VISUAL OBJECT TRACKING BY USING DEEP NEURAL NETWORKS

Hasan SARİBAŞ

PhD Dissertation

Department of Avionics

Supervisor: Asst. Prof. Dr. Sinem KAHVECİOĞLU

(Co-Supervisor: Prof. Dr. Hakan ÇEVİKALP)

Eskişehir

Eskişehir Technical University

Institute of Graduate Programs

August 2020

This thesis was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant number EEEAG-116E080.

ABSTRACT

VISUAL OBJECT TRACKING BY USING DEEP NEURAL NETWORKS

Hasan SARİBAŞ

Department of Avionics

Eskişehir Technical University, Institute of Graduate Programs, August 2020

Supervisor: Asst. Prof. Dr. Sinem KAHVECİOĞLU

(Co-Supervisor: Prof. Dr. Hakan ÇEVİKALP)

In this thesis, new methods have been proposed to track arbitrary objects. To this end, first, a novel lightweight two-stream deep learning-based tracker has been developed to obtain state-of-the-art results on different single object tracking benchmarks. In the proposed deep learning-based method, both spatial and temporal features are used and a ranking loss is employed. Using ranking loss term in the optimization function enforces the neural networks to learn to give higher scores to the candidate regions that better frame the target than the regions that frame the target object with less accuracy. The classifier scores received from spatial and temporal networks are fused. The experiments were conducted on eight different benchmarks. The proposed tracker achieves state-of-the-art results on most of the tested challenging tracking benchmarks. In addition to a deep learning-based tracker, a hybrid correlation filter-based tracker has been developed to detect and track UAVs. To detect the target UAV at the beginning of the tracking and in the case where the tracked UAV has been lost, the deep learning based YOLOv3 detector has been used. A kernelized correlation filter has been used to track detected target UAVs in real-time. Combining the detector and tracker provides high accuracy and real-time performance even on onboard computers. A new dataset called as UAV Tracking has been created to train the YOLOv3 detector and test the proposed method. The proposed method achieves state-of-the-art performances on created UAV Tracking dataset. Finally, the proposed method has been generalized for general tracking.

Keywords: Object tracking, Deep learning, Ranking loss, Correlation filter, Unmanned aerial vehicles.

ÖZET

DERİN SİNİR AĞLARI KULLANILARAK GÖRSEL NESNE TAKİBİ

Hasan SARİBAŞ

Havacılık Elektrik ve Elektroniği Anabilim Dalı

Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Ağustos 2020

Danışman: Dr. Öğr. Üyesi Sinem KAHVECİOĞLU

İkinci Danışman: Prof. Dr. Hakan ÇEVİKALP

Bu tezde, nesne takibi için yeni yöntemler önerilmiştir. İlk olarak, farklı nesne takip veri setlerinde en iyi sonuçları elde edebilmek için derin öğrenme tabanlı 2 akışlı küçük mimariye sahip bir yöntem önerilmiştir. Önerilen yöntemde iki akışlı bir yapay sinir ağı ile hem uzamsal hem de zamansal bilgilerden yararlanılmış ve de ranking loss fonksiyonu kullanılmıştır. Önerilen ranking loss optimizasyon fonksiyonu ile sinir ağları hedef nesneyi daha iyi çevreleyen bölgelere daha yüksek skor vermeyi öğrenir. Uzamsal ve zamansal sinir ağlarından gelen skorlar birleştirilir. Önerilen yöntem test edilen nesne takip veri setlerinin çoğunda en iyi sonuçları elde etmiştir. Derin öğrenme tabanlı yönteme ek olarak, İnsansız hava araçlarının (İHA) tespit ve takibi için ilinti filtresi tabanlı hibrit bir yöntem önerilmiştir. İHA'ların ilk framede ve takipçinin başarısız olduğu durumlarda tespit edilmesi için derin öğrenme tabanlı YOLOv3 nesne tespit yöntemi kullanılmıştır. İHA'ların gerçek zamanlı olarak takip edilmesi için ise Çekirdekleşmiş İlinti Filtresi kullanılmıştır. Nesne tespit ve takip yöntemlerinin birlikte kullanılmasıyla, tek kart bilgisayarlarda bile gerçek zamanlı çalışabilen yüksek doğruluğa sahip yöntem geliştirilmiştir. YOLOv3 nesne tespit yönteminin eğitilmesi ve önerilen yöntemin test edilmesi için UAV Tracking adında yeni bir nesne takip veri seti oluşturulmuştur. Önerilen yöntem bu veri set üzerinde en iyi sonuçları elde etmiştir. Son olarak, önerilen yöntem genel nesne takibi için genelleştirilmiştir.

Anahtar Sözcükler: Nesne takibi, Derin öğrenme, Ranking loss, İlinti filtresi, İnsansız hava araçları.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Asst. Prof. Dr. Sinem KAHVECİOĞLU for her dedicated support and guidance and encouragement throughout this dissertation study.

I would like to express my sincere gratitude to my co-supervisor Prof. Dr. Hakan ÇEVİKALP for his advice, valuable guidance and overall insights in this field have made this an inspiring experience for me.

I would like to thank the members of the dissertation monitoring committee, Ass. Prof. Dr. Helin DUTAĞACI and Ass. Prof. Dr. Hakan KORUL for their time, insightful comments, advices, supports and improvements.

I would like to thank the other members of the doctoral thesis defense jury, Prof. Dr. Ömer Nezih GEREK, Assoc. Prof. Dr. Emre KIYAK and Ass. Prof. Dr. Hasan Serhan YAVUZ for their relevant questions and comments.

I would like to thank the Scientific and Technological Research Council of Turkey (TUBITAK) BIDEB 2211-E program.

This dissertation was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant number EEEAG-116E080.

I wish to thank my wife Sevcan SARİBAŞ for her dedicated patience, limitless love and supporting me during the compilation of this dissertation. I also thank mom, my brother, and my sister for always believing in me and encouraging me.

Finally, I would like to thank my deceased father Süleyman SARİBAŞ for his valuable guidance, profound belief in my studies and abilities. I dedicate this thesis to him…

Hasan SARİBAŞ

05/08/2020

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with "scientific plagiarism detection program" used by Eskişehir Technical University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Hasan SARİBAŞ

CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF IMAGES

Page

GLOSSARY OF SYMBOLS AND ABBREVIATIONS

AO	:	Average Overlap
AUC	:	Area Under Curve
CF	:	Correlation Filter
CNN	:	Convolutional Neural Network
conv	:	Convolutional
CPU	:	Central Processing Unit
DCF	:	Discriminative Correlation Filter
DFT	:	Discreate Fourier Transform
EAO	:	Expected Average Overlap
ECO	:	Efficiently Convolutional Operators
fc	:	Fully Connected
FPS	:	Frame per Second
GPU	:	Graphics Processing Unit
HOG	:	Histogram of Oriented Gradients
IoU	:	Intersection over Union
KCF	:	Kernelized Correlation Filter
MDNet	:	Multi-Domain Convolutional Neural Network Tracker
NMS	:	Non-Max Suppression
OPE	:	One-Pass Evaluation
pool	:	Pooling
Pre	:	Precision
Pre_{norm}	:	Normalized Precision
ReLU	:	Rectified Linear Unit
ROI	:	Region of Interest
RPN	:	Region Proposal Network
SR	:	Success Rate
SVM	:	Support Vector Machine
UAV	:	Unmanned Aerial Vehicle
YOLO	:	You Only Look Once

1. INTRODUCTION

This thesis is focused on tracking the objects in images captured by unmanned aerial vehicles (UAVs). To this end, two different trackers, deep learning-based and CF-based, have been proposed. The goal of visual object tracking is to track an unknown object, which is specified in the initial frame mostly by a bounding box, in the successive frames. It is one of the computer vision's most widely studied problems and has many applications in many areas such as robotics, surveillance, activity and video analysis, human-machine interactions, etc. Despite considerable research efforts, tracking of the objects remains a challenging problem as target objects often undergo significant changes in appearance due to abrupt motion, occlusion, deformation, cluttered background and pose variation. Moreover, abrupt and severe changes in illumination as well as presence of similar objects around the object being tracked make the problem even more difficult. The main challenge arises from the fact that only the target object's bounding box is known in the first frame and generally no other prior knowledge or appearance model is available. The tracker has to adapt to all variations of the tracked object in the subsequent frames using a very limited information; thus, most methods easily drift away from the target object.

To compare with other state-of-the-art trackers, a general object tracking algorithm has been proposed. The proposed methods have also been tested on visual object tracking benchmarks which consisted of images captured by UAVs. The proposed convolutional neural network (CNN) based approach achieved state-of-the-art performances on different tracking benchmarks however, it does not run in real-time due to high computational complexity. To overcome this problem, a new correlation filter (CF) based method that can run in real-time has also been developed.

1.1. Motivation and Contributions

Deep learning-based trackers generally draw positive and negative samples around the estimated target location based on Intersection over Union (*IoU*) ratio and the model is trained by using these samples. To estimate the target location in the new frame, candidate samples are drawn around the estimated target position in the previous frame and the sample with the highest score is determined as the new location of the target. As a typical rule, when the *IoU* of the samples to be used to update the model are greater than 70%, they are selected as positive samples, while they are selected as negative

samples if the *IoU* of the samples are smaller than 30%. Generally, this sample selection strategy is not a good since it does not encourage achievement of sharper and more precise target localization. As a result, the learned model and the classifier are affected even by slight inaccuracies, and the trackers gradually drift and eventually fail to track the target object. On the other hand, the success of CFs is mostly due to the precise localization of the target center, but they are incapable of adapting to aspect ratio changes. This is owing to the efficient and approximate dense sampling performed by circularly shifting training samples. Because the circular shifts correspond to the actual translation of the target object location, this approach works well with homogeneous backgrounds and a generally stable target object. However, in addition to these boundary effects, a major problem with the CF-based trackers is their inability to estimate aspect ratios of the target object bounding boxes (To the best of our knowledge, [1] has used different CFs for the corners of the target object to address this problem). As a result, despite the precise localization of the target center, the rotating bounding box of the target object fails to be satisfactory, especially in the case of abrupt aspect ratio changes.

In this thesis, two different trackers, deep learning-based and CF-based, have been proposed to solve these problems. Deep learning-based approach has been employed to achieve state-of-the-art performance and CF-based approach to run in real-time onboard within the UAV.

The proposed lightweight two-stream deep learning-based tracker uses both spatial and temporal features. This tracker not only estimates the location of the target object but also fine-tunes the location and returns more precise bounding boxes that frame the target object by using a novel classification loss as well as simple but effective model update strategies during tracking. In more detail, the newly developed loss function consists of two terms: classical hinge loss and ranking loss. The hinge loss separates the positive sample instances from the negative sample instances, while the ranking loss returns a more precise location of the target object. [2] has shown that the ranking loss also improves accuracy in the object detection problem as expected. Besides, the ranking loss part can be interpreted as the distance similarity metric learning used in Siamese networks. In addition to the loss function, a simple yet effective update rule, which is different from the state-of-the-art tracker MDNet (Multi-Domain Convolutional Neural Network Tracker) [3], has been developed in the proposed method. During the online

tracking stage, two different models have been defined: a *current model* which is constantly updated and a *cache model* which is updated only at regular intervals. The current model is updated when scores of both the current model and the cache model are larger than the predefined threshold. The cache model is updated when there is a sample with a successful score at every 10 frames. This strategy is different from the updating strategy of the MDNet where the model is updated only at some pre-defined intervals (short-term and long-term) or whenever potential tracking errors are detected. The experiments have shown that both proposed novel loss term and updating strategy improve tracking accuracy and robustness under challenging conditions such as scale/aspect ratio changes, appearance changes.

Briefly, the contributions of the thesis to deep learning-based tracker can be outlined as follows:

- A novel lightweight two-stream deep learning-based method that uses both appearance and motion information for object tracking is introduced.
- The proposed approach employs a novel ranking loss function, which forces the tracker to return more precise bounding boxes that frame the target, to successfully adapt to scale/aspect ratio and appearance changes in target object.
- A cache model is employed to increase robustness performance under occlusion.
- A new update strategy that is compatible with the proposed ranking loss function is proposed.

In addition to deep learning-based tracker, a CF-based hybrid method is presented for tracking objects in the images captured by UAVs. This approach follows a tracking by detection strategy. YOLOv3 (You Only Look Once) and YOLOv3-Tiny [4] models, which provide one of the best trade-offs between speed and accuracy in the literature, are used to detect the target object in the beginning and in case the tracked target is lost. A kernelized correlation filter (KCF) [5] is utilized to track this detected target object in real-time. Combining these two approaches ensures high speed and accuracy even on onboard computers such as JetsonTX2. The experiments have shown that the proposed CF-based hybrid trackers obtained state-of-the-art performance compared to real-time trackers on different datasets.

In the following sections, firstly, related works for a single object tracking are reviewed. The methodologies are given in the third section, and proposed components are discussed in this section. In the fourth section, we conduct experiments, compare our approaches with other state-of-the-art trackers, and we also discuss improvements of our proposed approaches. Finally, evaluation of the results of different proposed trackers is given in the conclusion section.

2. RELATED WORK

Earlier visual object tracking studies typically belong to one of two categories: generative trackers and discriminative trackers. Generative tracking methods are built based on the assumption that target appearances can be represented by using generative models; therefore, these approaches search for the target regions that best fit those generative models around the previous location of the target object. Some representative methods employ dictionary learning [6], subspace learning [7, 8], density estimation [9, 10], regression networks [11], and sparse representation [8, 12, 13]. On the opposite, discriminative methods address the problem as binary classification and learn a classifier to separate the target object from the background. To this end, various classifier approaches such as online boosting [14], ensemble of classifiers [15, 16], linear support vector machine (SVM) [17], and multiple instance learning [18] were used.

Discriminative methods have higher accuracy scores compared to generative methods. However, they also have weaknesses such as slow runtime, ability to track only target objects for which they were trained, failure to recognize the target object in case of appearance changes unless the model is updated compared. A more complete list of previous single object tracking methods and comparisons are presented in [19, 20].

Most of the state-of-the-art object tracking studies carried out in recent years can be roughly categorized under three groups: Deep learning-based trackers, Siamese network-based trackers and correlation filter-based trackers.

2.1. Deep Learning-Based Trackers

CNN based trackers usually use lightweight deep neural networks as given in [3, 15, 18, 21-24] because the last layers of large CNNs are more effective in capturing semantics. Hence, they are suitable for object classification and detection problems but not for tracking problem. The earlier layers of CNNs are more sensitive in capturing spatial details which are more important for object tracking problem. In addition, small networks are required because of speed issues. At online updating stage, updating parameters of large networks are slow since there are many parameters that cannot be performed in real-time for tracking. Therefore, the usage of small networks is more common in object tracking problem. Generally, both small and large networks update either a few last layers or the last classifier layers of the network during online learning stage for the sake of computational efficiency.

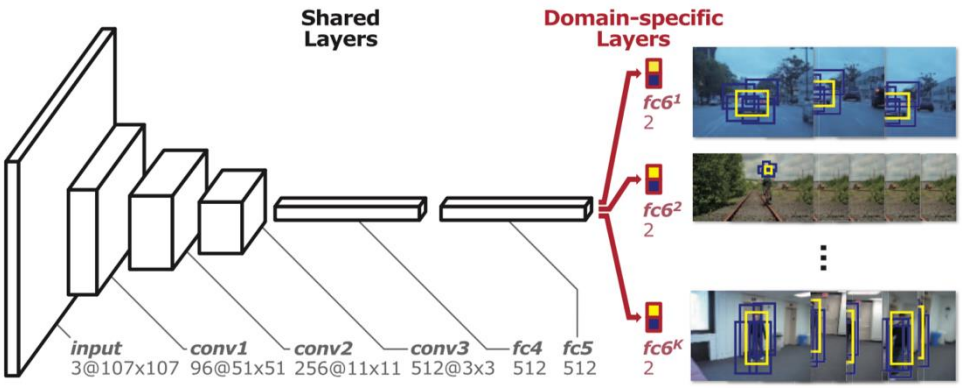


Figure 2.1. *MDNet architecture [3]*

Nam and Han (MDNet) [3] were inspired by R-CNN (Regions with CNN Networks) [25], which is an object detection algorithm, to solve object tracking problem. Figure 2.1 illustrates the architecture of MDNet which is a very well-known deep learning-based tracker. They filled in the gap between detection and tracking using a multi-domain network in the last classification layer during offline training process. This layer provides a class agnostic structure since it creates a domain-specific classifier which changes according to video number. It uses part of VGG-M (Visual Geometry Group) [26] network which consists of 3 convolutional layers and 3 classification layers. This approach includes offline and online training stages. In the offline training stage, positive and negative samples are chosen around the ground-truth from the input image consistent with IoU between ground-truth and selected samples. These samples are cropped from the input image and reshaped to $107x107$ size then classified using domains specific layers. In the online learning, positive and negative samples are again chosen around ground-truth of the first frame and only fully connected layers of pre-trained network are trained. In addition to initial training, long-term and short-term updates were also used in this approach. This architecture is also used as a baseline in this thesis. Real-Time MDNet [27], which is a fast version of MDNet tracker, proposed real-time object tracking method inspired by Fast R-CNN [28] object detection method which uses accelerating feature extraction procedure. This method improved the RoiAlign technique and discriminative embedding loss, thus allowing more discriminative features for object classification to be learned. A stacked denoising auto-encoder was used by one of the earliest deep learning-based trackers [29] to learn generic image features as auxiliary data from a large dataset,

and then learned features were transferred to online object tracking. In offline stage, Wang et al. [30] pre-trained a large CNN layers, and transferred the learned features to the online tracking as in [29]. Hong et al. [31] used CNN layers and SVMs to learn target-specific saliency maps. Li et al. [32] developed an online learning strategy based on a CNN using multiple image cues.

2.2. Correlation Filter-Based Trackers

On the other hand, CF-based trackers learn a filter to locate the target object accurately in consecutive frames by solving computationally efficient ridge regression problem in the Fourier frequency domain. In the new frame, the learned correlation filter is applied to Region of Interest (ROI) that generally chooses around 2 or 3 times larger than bounding boxes of the target object, and the maximum correlation filter response is used to determine the location of the target object. Correlation filter-based trackers are extremely fast compared to the deep learning-based trackers because the problem is solved in the frequency domain. Earlier CF-based trackers used hand-crafted histogram of oriented gradients (HOG) [5], color histograms [21], color names [33] or gray levels features. Then studies [34, 35] utilized pre-trained CNN features to improve accuracy. In the last few years, most techniques [36-38] based on correlation filters employed new architecture which can learn both correlation filters and CNN features. In object tracking, the first competitive results have been obtained with CF by MOSSE (Minimum Output Sum of Squared Error) [23] tracker which uses the minimum output of squared error to solve the problem. Figure 2.2 shows examples of correlation filter-based techniques that use different formulations to obtain the correlation filter. As seen in the figure, the filter and the output of the MOSSE filter are more robust than other trackers including Naive, UMACE, and ASEF filters. Kernelized correlation filter [5] has been proposed using circulant matrices and multi-channel features. Danelljan et al. [39] proposed a theoretical formulation which enables the integration of multi-resolution deep feature maps into the correlation filter-based trackers to learn discriminative convolutional operators. The formulation achieved high accuracy scores but was not as fast as other CF-based trackers. To solve speed issue efficiently convolution operators (ECO) have been further improved by introduction of factorized convolutional operators in [24].

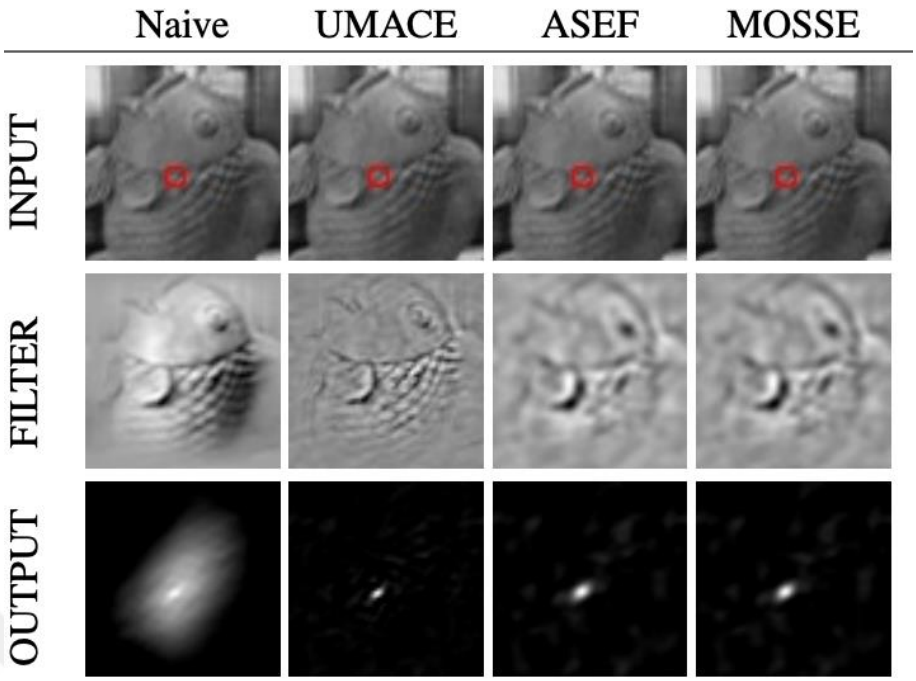


Figure 2.2. *Comparison of correlation filters [23]*

In the Visual Object Tracking 2017 (VOT-2017) [40] challenge, the winner algorithm [38] utilized a ROI based pooling operation in the correlation filters. Wang et al. [41] introduced an unsupervised tracking algorithm based on forward-backward trajectory analysis to improve the object tracking accuracy. Xu et al. [42] introduced a new group feature selection (GFS) method to reduce both spatial and channel dimensions. Dai et al. [43] introduced adaptive spatially regularized CFs (ASRCF) to learn an effective spatial weight for a specific object. Huang et al. [44] proposed a novel discriminative correlation filter approach capable of effectively and efficiently eliminating aberrances to solve undesired boundary effects. In addition to these CF trackers, different CF methods have been proposed to increase tracking accuracy and robustness can be accessed in [45, 46].

2.3. Siamese Based Trackers

In addition to deep learning-based approaches which used detection by tracking, and CF-based approaches, some studies formulated object tracking problem as a one-shot detection task [22, 47, 48]. Tao et al. [47] introduced the first Siamese based object tracker which simply finds the target object in the following frames using a matching function which was learned from large datasets such as ILSVRC 2015 (ImageNet Large Scale

Visual Recognition Competition) [22] proposed a fully-convolutional Siamese (SiamFC) network to track arbitrary objects using similarity learning. Figure 2.3 shows that SiamFC architecture is fully-convolutional with respect to the candidate image *x*. This network was trained to learn embedded function using positive and negative image pairs on a large dataset.

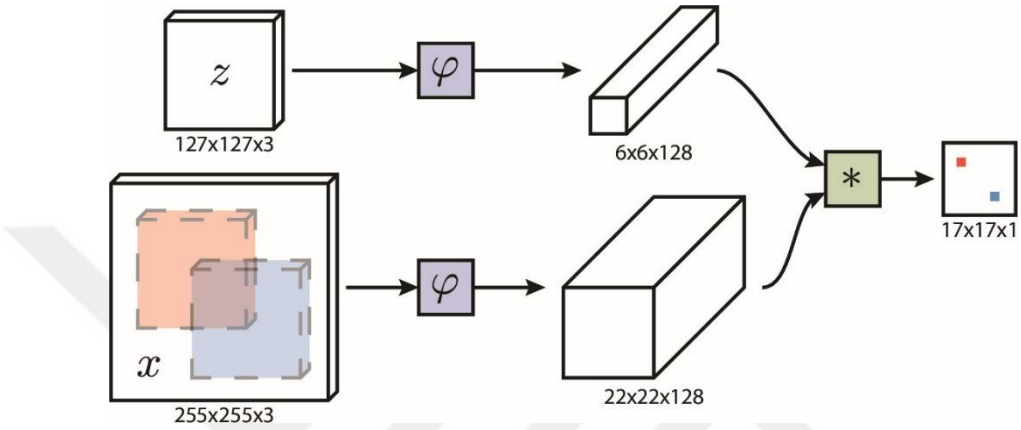


Figure 2.3. *Fully-convolutional Siamese network for object tracking [22]*

This method learns a similarity metric using only the first frame and does not have the ability to update this metric in the next frames during tracking thus it is capable of real-time tracking. To solve updating similarity metric, Wang et al. [48] and Guo et al. [49] proposed novel Siamese-based approaches. Initially, Siamese-based trackers were not capable of adapting to aspect ratio changes. SiamRPN [50] used region proposal networks (RPN) in object tracking to adapt aspect ratios changes when determining the location of target objects in the candidate image. To reduce the model size and to improve accuracy and robustness, SiamRPN++ [51] introduced a layer-wise feature aggregation structure. Fan and Ling [52] employed cascaded RPN and hard negative sampling process instead of one-stage RPN to solve the data imbalance problem in RPN. GradNet [53] proposed a novel gradient-guided network in object tracking to increase adaptation ability and avoid over-fitting. SiamMask [54] proposed a novel Siamese-based architecture for both object tracking and segmentation problems simultaneously using a multi-task learning approach. To improve accuracy, Choi et al. [55] proposed a novel meta-learner by adding target aware feature space for tracking. SPM-tracker [56] utilized a two-stage series-parallel matching where fine and coarse matching (FM-CM) stages increase the discriminative power and robustness of Siamese trackers.

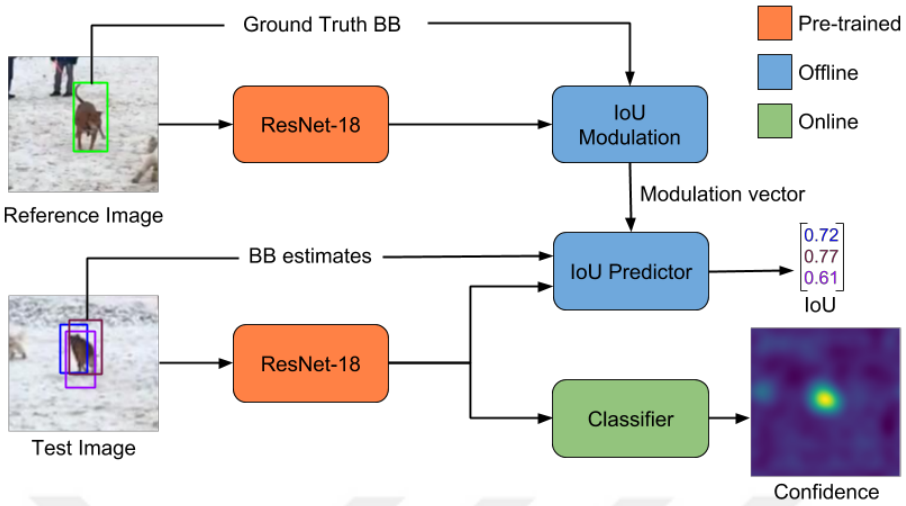


Figure 2.4. *Overview of ATOM tracker architecture [57]*

In more recent years, ATOM [57] addressed two different components of the object tracking problem: estimation module and classification module. It used the discriminative correlation filter (DCF) and Intersection over Union network (IoU-Net) for classification and estimation, respectively. Figure 2.4 shows the overview of ATOM tracker network, the blue blocks, which are the target estimation module, are trained offline on large scale datasets and the green block, which is the classification module, is trained online. The winner of the VOT 2019 [58] short-term challenge integrated ATOM and SiamMask trackers to obtain high accuracy. Bhat et al. [59] introduced DiMP (Discriminative Model Prediction) tracker. DiMP used both estimation and classification modules to increase robustness performances of the trackers. It also developed a target model prediction mechanism which is derived from a discriminative learning loss to obtain the best target model using previous models. Figure 2.5 shows the overview of DiMP tracker classification module network. The architecture of this method is trained offline in end-to-end manner. Huang et al. [60] developed a first general framework for object tracking approach on deep learning-based detectors using a target-guidance module and a meta-learner to bridge the gap between detection and tracking.

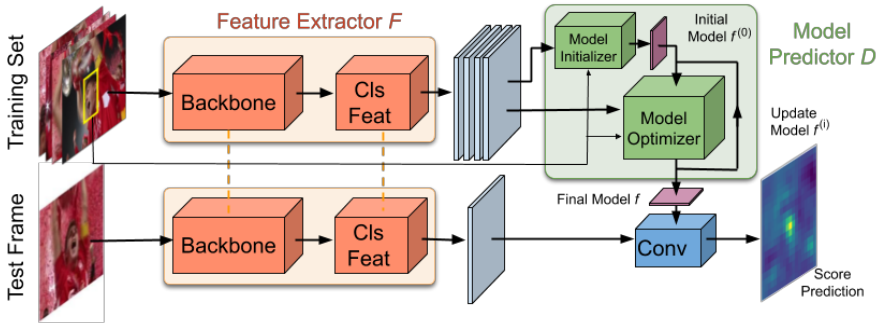


Figure 2.5. *Overview of DiMP tracker classification module [59]*

Video frames consist of spatial and temporal information. To the best of our knowledge, generally, deep learning-based trackers have been used spatial features, which are obtained from RGB channel in video frames, for object tracking. Both spatial and temporal features have been widely utilized by [61-65] for action recognition and detection problems. Simonyan and Zisserman [61] used a two-stream CNN network that employs both RGB and optical flow frames to achieve high performance in action recognition. Similarly, for action localization, [62] fused spatial and motion CNN features and [63] used a two-stream 3D CNN features extracted from optical flow and RGB networks. [64] utilized spatial-temporal features to learn similarities between video streams. Inspired by their success in action localization and recognition problems, some recent tracking methods [66-70] have begun to utilize the spatial-temporal features. However, as we do not have access to all video frames during online object tracking, this is not as straightforward as is in action recognition. In object tracking, we have some disadvantages; we only have access to the previous frames from the current frame and some tracking videos do not have any motion. As a result, special attention must be given to integration of spatial and motion information during online tracking. As a matter of fact, temporal information is mostly useful under several factors such as illumination variations, out-of-plane rotations, and appearance changes in object tracking. If the target object is moving in video stream, motion information also helps. In the context of tracking, [66] fused hand-crafted, deep RGB, and deep motion (optical flow) features in a tracking-by-detection framework to show that deep motion features increased the tracking performance. FlowTrack [67] developed an end-to-end flow discriminative correlation filter-based tracking method using a novel spatial-temporal attention mechanism to improve tracking accuracy and the feature representation. [68] introduced a novel deep architecture that incorporates global and local, spatial and temporal

information of the target object to obtain high-performance tracking results. Also, a temporal network using an online tuple learning process is proposed to solve the drifting problem in [68]. Gao et al. [69] proposed Siamese based end-to-end spatial-temporal graph convolutional tracking (GCT) framework for object tracking.

3. METHOD

Two different methods have been proposed for object tracking within the scope of the thesis. The first one is the deep learning-based method which is developed for general single object tracking applications. This approach shows state-of-the-art performance but is slow. This deep learning-based method runs at ~3-4 FPS on a single 16 GB Quadro P5000 GPU server. Because of the speed problem, it is not possible to operate on unmanned aerial vehicles in real-time. The second one is a real-time method that can run on the Jetson TX2 onboard computer, can be integrated within an unmanned aerial vehicle. Details of both methods are given below.

3.1. Deep Learning-Based Method

Video sequences include spatial and temporal information. In the video frames, the spatial component given in the form of individual frames carries information about the scenes and appearances of objects. On the other hand, the temporal component given in the form of motion over the frames captures object movement. To take the advantage of both spatial and temporal information, the thesis proposes a lightweight two-stream deep learning-based tracker for object tracking. In the proposed architecture, the spatial network (RGB Network) employs RGB frames to capture the appearance of the target object, while the temporal network (Optical Flow Network) employs optical flow features to capture the motion of the target object.

The x and y components of the motion vectors are fed to the optical flow network input by stacking, and these values are converted linearly to integers between 0 and 255. Both RGB and Optical Flow networks are structurally the same, and each network consists of 3 convolutional layers and 3 fully connected layers. The classification scores that come from the two streams are fused for final inference at a late stage of the network. To classify candidate samples, we propose a new classification loss function as opposed to the classic soft-max loss used in many deep neural networks. The proposed loss function is more robust to potential drifts from the target objects and more suitable for object tracking.

The MDNet architecture is used as a baseline. Our one stream network contains 3 convolutional (*conv1-3*) layers which are the first three layers of VGG-M network (shared layers) followed by 3 fully connected (*fc4-6*) layers.

Table 3.1 indicates the proposed tracking algorithm during online tracking stage. Especially, this algorithm focused on the proposed online updating rule and usage of the active and cache models. In this algorithm, t is the frame number, x_t denotes the location of the target at t th frame, $f(x_t)$ is the classifier score of the x_t location.

Table 3.1. *Algorithm of the proposed tracker*

	Algorithm
1:	Around x_1 S_1^+(positive), S_1^- (negative), S_1^R (ranking) samples are selected
2:	Determine the active and cache model.
3:	Update {*fc4, fc5, fc6*} using S_1^+, S_1^-, S_1^R
4:	**Repeat**
5:	Determine the candidate bounding boxes $x^i_{(t,temporal)}$.
6:	[$f(x_{(t,temporal)})$ $x_{(t,temporal)}$] = find($max(f(x^i_{(t,temporal)}))$).
7:	Determine the local candidate bounding boxes $x^i_{(t,local)}$ very close to $x_{(t,temporal)}$.
8:	[$f(x_{(t,local)})$ $x_{(t,local)}$] = find($max(f(x^i_{(t,local)}))$).
9:	**if** $f(x_{(t,local)}) > f(x_{(t,temporal)})$ **then**
10:	$x_t = x_{(t,local)}$.
11:	$f(x_t) = f(x_{(t,local)})$.
12:	**else**
13:	$x_t = x_{(t,temporal)}$.
14:	$f(x_t) = f(x_{(t,temporal)})$.
15:	**if** $f(x_t)$ **> (Threshold) then**
16:	Update {*fc4, fc5, fc6*} using S_1^+, S_1^-, S_1^R.
17:	**else if** $f(x_t)$ **< 0 then**
18:	Update {*fc4, fc5, fc6*} using Cache model.
19:	**if mod(t,10) == 0 &&** $f(x_t) > 0$ **then**
20:	Update cache model.
21:	**if** $f(x_t)$ **< -0.5 then**
22:	$x_t = x_{t-1}$.
23:	$t = t + 1$.
24:	**Until** (end of the video sequence)

For offline learning stage, the same strategy is utilized as in MDNet. In more detail, to provide class-agnostic classification, we treat each training video stream as a different domain and train the network with K branches of training videos - the domain-specific layers. To this end, the method uses the same network which consists of the first three convolutional layers and 2 fully connected layers for each video sequence in the offline training stage, however, the last fully connected layer (fc6) changes in each video sequence. In online tracking stage, we employed only shared layers (*conv1-3*) which are pre-trained in offline learning and also used a completely different model update strategy compared to MDNet.

We will start by explaining the CNNs, then the proposed new loss function before we describe the overall tracking stages with the fusion strategies that are employed to combine two-stream networks. This will be followed by a description of the model update tricks used during online tracking. Finally, we will explain how to set the parameters of the proposed deep neural network architecture.

3.1.1. Convolutional neural networks (CNNs)

CNNs generally contain a sequence of different layers including convolutional (*conv*) layer, pooling (*pool*) layer and fully connected (*fc*) layer and rectified linear unit (ReLU). *conv* and *fc* layers have parameters, while the other layers have no parameters. Figure 3.1 shows a simple CNNs architecture.

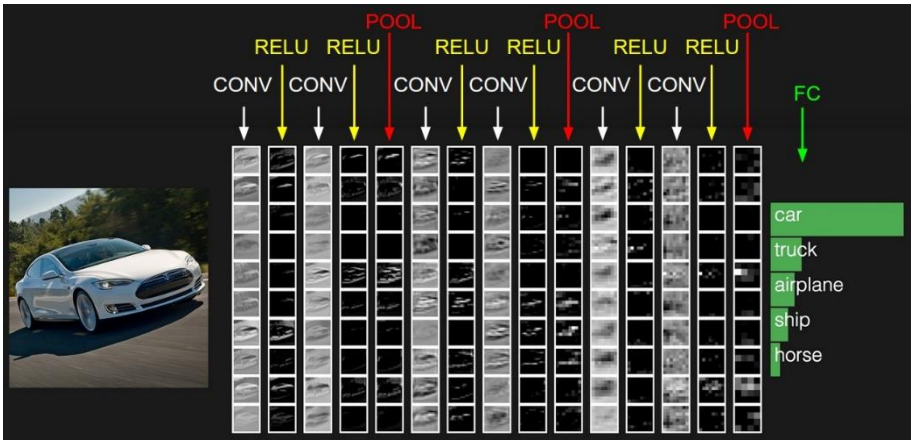


Figure 3.1. *The overview of the tiny VGG Net architecture [71]*

3.1.1.1. Convolutional layer

The *conv* layer having most of the parameters of the network, is the main building block of the CNNs. The parameters of the *conv* layer contain a set of learnable filters. These filters extend across the full depth of the input volume, but they have small width and height. Using this operation, different features are obtained from the input matrix. To this end, we slide filters across the width and height of the input matrix. Stride determines how the filter convolves across the input volume. Stride is also directly related to the output volume, for example when we use stride 1, the output and input volume will be of the same size. If the stride increases, the output volume reduces.

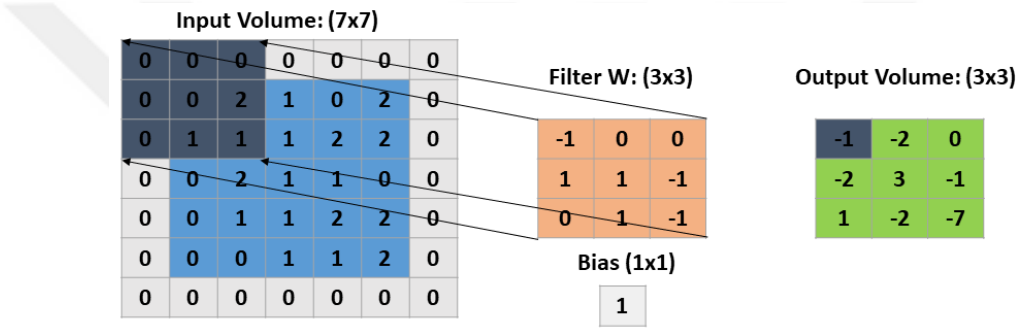


Figure 3.2. *2D Convolutional operation*

Output volume is calculated by using the equation (3.1).

$$Output\ Volume = \frac{W - F + 2P}{S} + 1 \tag{3.1}$$

where W is the input volume size, P is the padding, F is the filter size and S is the stride. For Figure 3.2 above, for a $5x5$ input and a $3x3$ filter with stride 2 and padding 1, we would get a $3x3$ output volume ($\frac{(5-3+2)}{2} + 1$).

3.1.1.2. Pooling layer

Generally, *pool* layer is inserted between two convolutional layers. Its function is to gradually reduce the spatial size of the representation to reduce the number of parameters and calculations in the network and thereby control overfitting. This layer operates independently in each depth segment of the input and resizes it spatially using Average and Max operations. Figure 3.3 shows the Max and Average pooling operations for the given input. In this figure, the input volume of size 4x4 is pooled with filter size 2

and stride 2 into output volume of size $2x2$. In this operation, the input volume is downsampled; however, the depth is preserved.

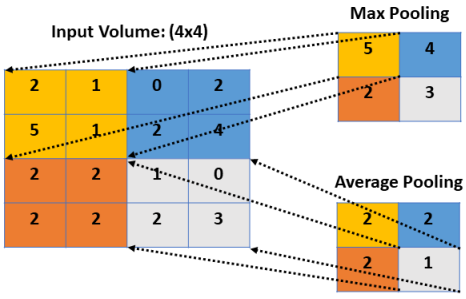


Figure 3.3. *Average and Max pooling operations*

3.1.1.3. Fully connected layer

The *fc* layer consists of a series of neurons, which are fully connected to all neurons in the previous layer as in classical neural networks. In a single layer, the neurons operate entirely independently and do not share any connections. The last *fc* layer is the output layer of CNNs and it represents output scores. Figure 3.4 indicates typical fully connected neurons for *fc* layers. The two neurons in the last layer represent the outputs of the neural networks.

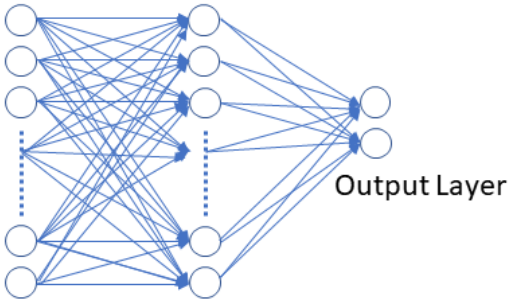
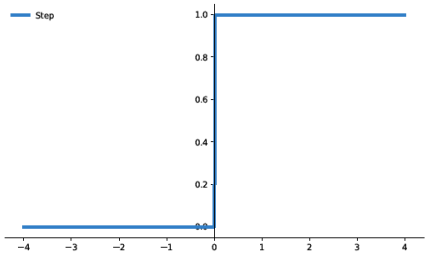
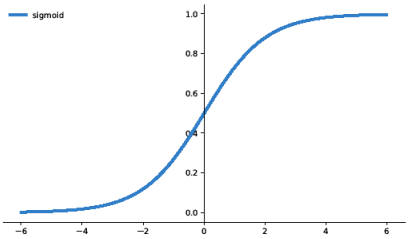
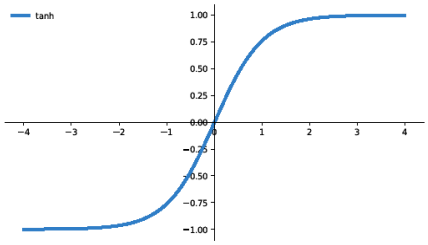
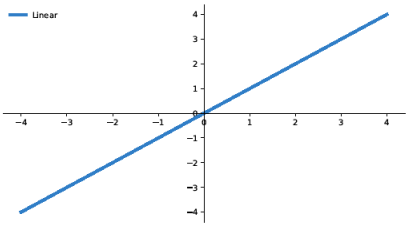
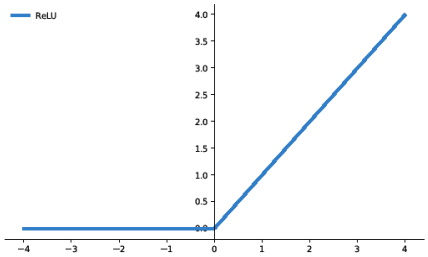


Figure 3.4. *fc layers*

3.1.1.4. Activation functions

By using $Y = \sum_i (Weight * inputs + bias)$ formulation, artificial neurons calculate a weighted sum of its inputs and add a bias. The range of output values of the neurons change from (-∞) to (∞). The neural network uses activation functions such as step function, sigmoid function, hyperbolic tangent (tanh) function and rectified linear unit function (ReLU) function to limit or activate this output as it happens in the brain. Table 3.2 shows the properties of several activation functions.

Table 3.2. *Equations and plots of activation functions*

Function name	Equation	Plot
Step	$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	
Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$	
Tanh	$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	
Linear	$f(x) = x$	
ReLU	$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	

3.1.2. Robust ranking loss

The deep learning classifier aims to find the best location of the target in the given video frame. Existing deep learning-based object tracking algorithms select positive and negative samples considering the *IoU* ratios around the estimated location of the target object and update the model using these samples. To update the model, samples with *IoU* ratios greater than 70% are selected as positive, and samples with ratios smaller than 50% to 30% are selected as negative. This method is not a suitable strategy to determine the target's location precisely and sensitively; therefore, the drift occurs between the target and the estimated location and the positive and negative samples selected around this drifted location are considered correct. Then the model is trained again using these incorrect samples, and the tracker fails gradually. In addition to these positive and negative samples, ranking samples are selected considering *IoU* ratios.

3.1.2.1. Intersection over union

In the Figure 3.5, the ground-truth bounding box is drawn in red and the predicted bounding box is drawn in blue. The *IoU* also is known as the Jaccard index and calculated by using the formulation, $IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$. In our figure, the area of overlap is shown in blue and area of union in yellow.

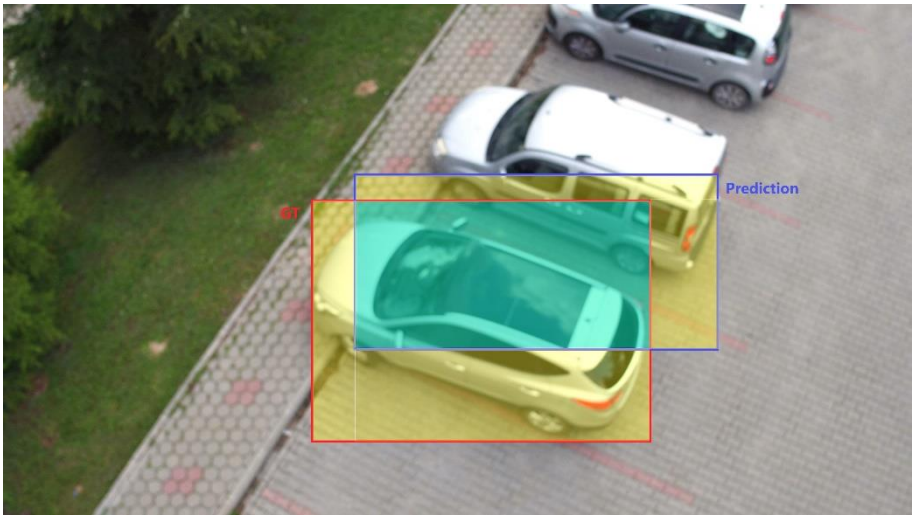


Figure 3.5. *Intersection over Union (IoU)*

These ratios play a crucial role in our loss function and *IoU* ratio is calculated using equation (3.2).

$$IoU = \frac{|B_{gti} \cap B_i|}{|B_{gti} \cup B_i|} \tag{3.2}$$

In (3.2), B_i and B_{gti}, ($i = 1,...,n$) represent the estimated location and ground-truth of the i th frame, respectively.

Let $x_{i,}$, ($i = 1,...,n$) represent the visual feature of a sampled region during online tracking and $y_{i,} \in \{-1, 1\}$ is the corresponding label. Our method utilizes the following loss function,

$$\underbrace{min_w \frac{\lambda}{2} trace(w^T w)}_{\text{Regularization}} + \underbrace{\sum_{r=1}^{n_r} L(r) H_1(w^T x_{r1} - w^T x_{r2})}_{\text{Ranking Loss}} + \underbrace{\kappa \sum_{i=1}^{n} H_1(y_i(w^T x_i))}_{\text{Hinge Loss}} \tag{3.3}$$

where $H_1(t) = \max(0, 1 - t)$ is the classical hinge loss, *L(.)* is a weighting function for different ranks and λ and κ are the parameters that must be set by the user.

In the optimization equation (3.3), the first term is the regularization term and various regularization methods have already been implemented in deep learning frameworks such as Pytorch, Tensorflow. Therefore, we will focus on the second and third terms in equation (3.3).

The last loss term in (3.3), which is the classical hinge loss, goals to separate the positive samples from the negatives, and this term ensures that the resulting classifier returns positive scores (≥ 1) for the positive samples and negative scores (≤ -1) for the negative ones. In fact, this term along with the first term in the optimization just implements a linear SVM classifier, and it is better suited for object tracking settings where there are very limited training samples. As given in the experiments, the tracker with the classifier using the first and the last terms performs even slightly better than MDNet which uses soft-max loss. This is quite natural since soft-max classifier is a probabilistic and nonlinear function which needs many training samples. The similar findings are obtained in [59], where the authors achieve better accuracies with linear SVM compared to soft-max classifier in the context of visual object detection. For labeling, the samples with *IoU* ratios greater than 0.7 are regarded as positive samples, and samples with *IoU* ratios smaller than 0.3 are treated as negative samples.

The second term in (3.3) is the major novelty in the proposed loss function, and it is specifically introduced for online object tracking. The main goal of this term is to ensure that the classifier gives higher scores to sampled regions that frame the target object better. To this end, the samples, with *IoU* ratios larger than or equal to 0.1 are used. As illustrated in Figure 3.6, randomly pairs are selected and the ones with a difference between *IoU* ratios greater than or equal to 0.4 are used. x_{r1} is the visual feature of the region that frames the target object better, and x_{r2} is the visual feature of the region that frames the object with less accuracy. If the score difference ($w^T x_{r1} - w^T x_{r2}$) is larger than 1, there is no loss; otherwise there exists a cost determined by *L(.)*. To compute *L(.)*, the following formula is used,

$$L(r) = [IoU(x_{r1}) - IoU(x_{r2})]\gamma \tag{3.4}$$

where $IoU(x_{r1})$ represents *IoU* ratio of the sample x_{r1}, and γ is a parameter that must be set by the user. Thus, *L(r)* value can be between 0.4γ and 0.9γ. It is observed that selecting hard x_{r2} examples (extracted from the regions that frame the tracked object with less accuracy) with high confidence scores improves overall accuracy around 1%. In each frame, the classifier learns to give higher scores for the sampled regions that frame the target object better since the classifier produces the highest score for the best target position. As a result, tracking failures caused by accumulation of errors due to the small drifts are largely reduced. There are other advantages of using this loss term: Although the context information is not aimed to be used directly, this term also takes advantage of context information since the samples with *IoU* ratios between 0.1 and 0.3 include regions mainly coming from the background. The proposed loss term can also be interpreted as a more systematic way of using distance learning controlled by the *L(.)* function. Therefore, the proposed tracker can also be regarded as a more advanced and specialized Siamese network tracker.

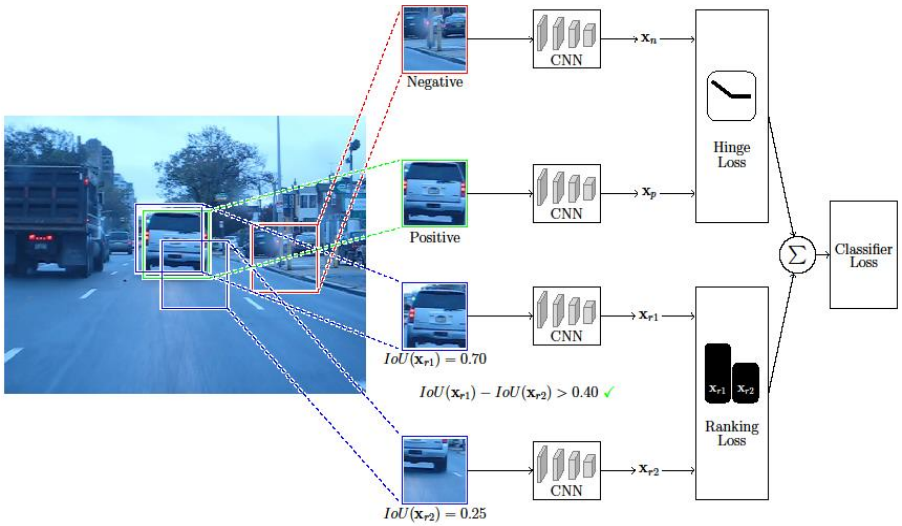


Figure 3.6. *Visualization of Ranking Loss and Hinge Loss Functions*

3.1.2.2. Hard ranking mining

Lastly, we would like to point out that similar procedures are used to improve the visual classification accuracy of deep neural networks through surrogate tasks by learning to solve jigsaw puzzles [72] or to sort sequences [73]. For example, in jigsaw puzzle solving, the image is divided into square patches in a grid, which are permuted randomly. Then, the task is to find the correct permutation that is going to restore the image. Samples of ranking pairs with high *IoU* ratios are sorted from the lowest to highest score, and in the opposite, samples of ranking pairs with low *IoU* ratios are sorted from the highest to lowest score. In other words, the ranking pairs were not matched randomly, and the loss was calculated between low-scoring samples with high *IoU* ratios and high-scoring samples with low *IoU* ratios. In this way, ranking samples were used more beneficially, and the performance of the method was improved as shown in the experiment. Table 3.3 shows 6 randomly selected ranking pairs and ranking loss are calculated only for 2nd and 5th pairs.

Table 3.3. *Samples are selected randomly*

IoU_H	IoU_L	$Scores_H$	$Scores_L$	Ranking Loss
0.9	0.14	3.2	2.1	-
0.7	0.25	2.8	3.0	+
0.8	0.42	1.5	0.1	-
0.74	0.18	4.2	0.53	-
0.79	0.33	1.7	1.4	+
0.84	0.38	4.5	3.4	-

Then, by applying hard ranking mining (Table 3.4), ranking samples are sorted by scores and the hard-ranking mining loss was calculated using three different pairs.

Table 3.4. *Samples sorted considering scores (applied hard ranking mining)*

IoU_H	IoU_L	$Scores_H$	$Scores_L$	Hard Ranking Loss
0.8	0.38	1.5	3.4	+
0.79	0.25	1.7	3.0	+
0.7	0.14	2.8	2.1	+
0.9	0.33	3.2	1.4	-
0.74	0.18	4.2	0.53	-
0.84	0.42	4.5	0.1	-

3.1.3. Optical flow

In general, the spatial information of the target object is used while tracking the object in video streams. In addition to this spatial information, as new frames are captured in the video streams, temporal information of the target object can be obtained using the previous frames. Optical flow or 3D-CNNs are generally employed to extract temporal (motion) features. 3D-CNNs are usually applied to video classification and action classification problems using stacked previous frames. However, due to the high computational cost of 3D-CNNs, we used optical flow to extract motion features. Optical flow calculates displacements of objects between consecutive frames. More precisely, optical flow determines the difference between the two frames by processing pixel wise and determines the displacement of pixels in two dimensions (x, y). In video frames, the optical flow produces meaningful features only if the target object is displaced; otherwise, it produces meaningless features. When the camera is moving the optical flows will be observed across the image; hence, the optical network will produce meaningless features. To utilize both spatial and temporal information, a two-stream architecture for object tracking was presented. While one network employs RGB frames to capture the target object, the other network employs optical flows to capture the motion of the target object. To calculate optical flows, the popular optical flow method [74] was used. Optical flow input is given as a 2-dimensional image by stacking the x-component and y-component of the motion vectors, and these values were linearly rescaled to integers between [0, 255]. Figure 3.7 shows some successful and failed examples of optical flows.

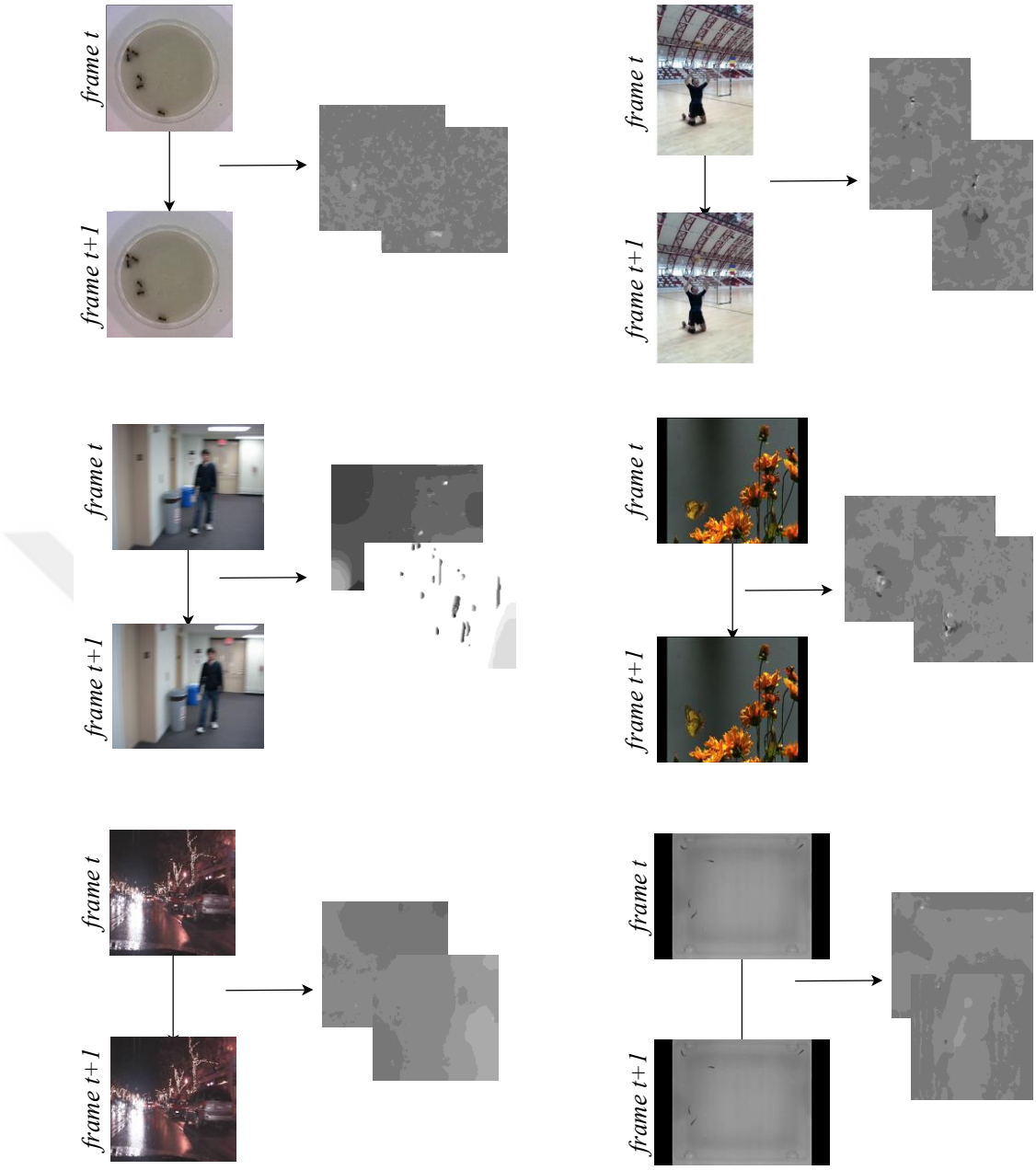


Figure 3.7. *Optical flow examples*

3.1.4. Fusion strategies for combining two-stream network

We propose two different strategies to combine two-stream networks: feature fusion (early fusion) and classifier output fusion (late fusion).

3.1.4.1. Feature fusion (early fusion)

We first employed the feature level early fusion method and simply concatenated 4608-dimensional RGB and optical flow CNN network features and then learned the classifier weights by using this combined representation as shown in Figure 3.8.

However, this strategy did not make any improvement over the single-stream method using only the RGB network. This is due to the fact that there was no movement in some video streams; therefore, the optical flow network did not produce any sensible output due to lack of motion. Therefore, when these insensible optical flow features were combined with RGB network features, they also reduced overall accuracy.

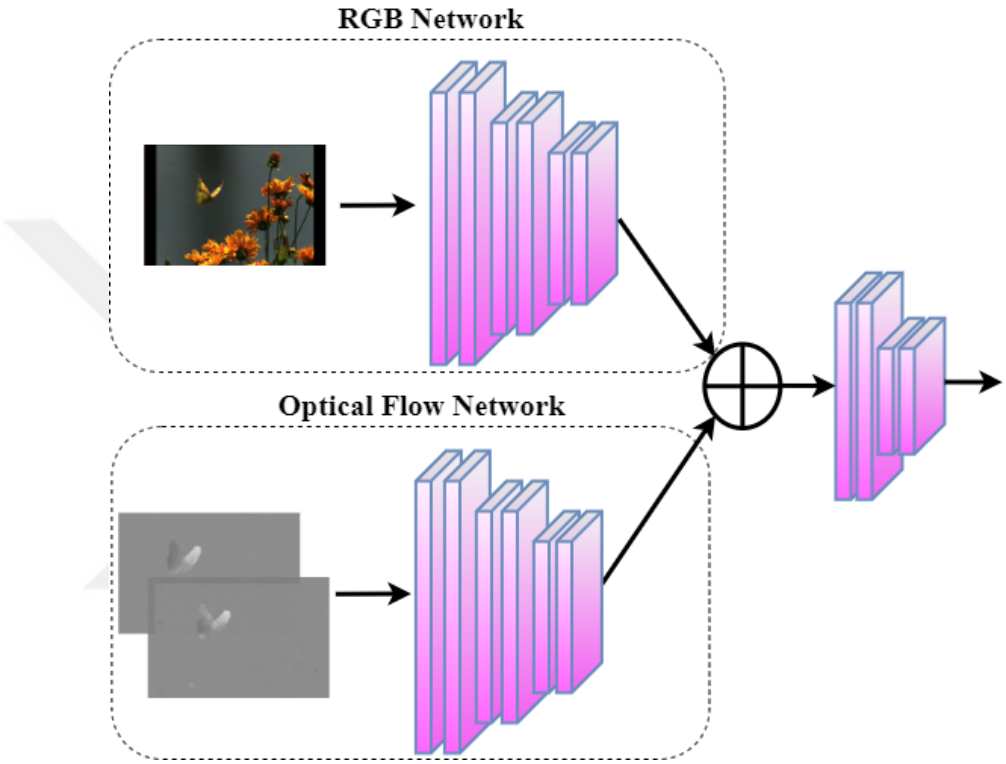


Figure 3.8. *Feature fusion (Early fusion)*

3.1.4.2. Classifier score fusion (late fusion)

As a second method, we decided to fuse classifier scores coming from the two networks. To fuse classifier outputs, we considered two approaches.

Average score fusion: In the first approach called RankingAF, we simply added classifier scores without introducing any weight. The designed network is shown in Figure 3.9. Experiments have shown that this strategy improves the performance of the method.

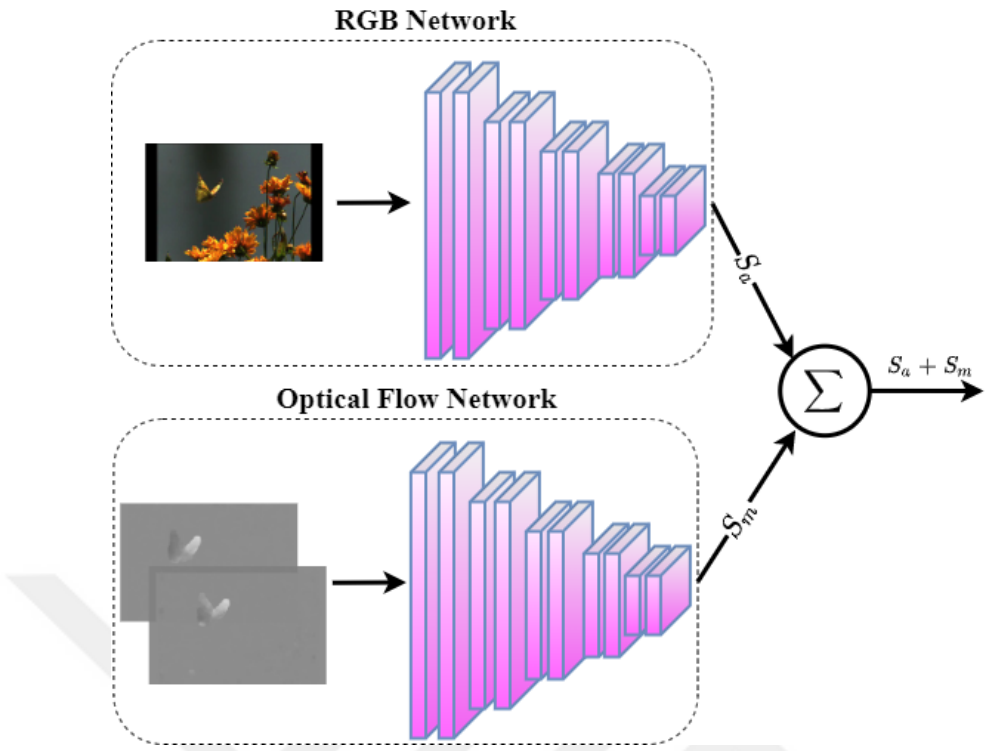


Figure 3.9. *Average score fusion*

Weighted score fusion: In the second approach called RankingSF, we learned class-specific weights by a layer in the network, then we used weighted sum of the classification scores for inference as illustrated in Figure 3.10. To achieve this, an end-to-end manner learnable network was designed to fuse scores using a layer with 2 parameters (w_a and w_b). Let S_a and S_m represent the classifier scores coming from RGB (appearance) and motion networks, respectively, and w_a and w_b be the corresponding class-specific weights that will be learned by the network. The total score is given as,

$$S = w_a S_a + w_b S_m \tag{3.5}$$

We enforced convex combination weighting restrictions on the learned weights such that they were positive and their sum was equal to 1, i.e., ($w_a, w_b \geq 0$), and ($w_a + w_b = 1$). To this end, these constraints were added to the loss function as given in (3.6). The RankingSF results are reported in the experiments section.

$$
\begin{aligned}
min_w \frac{\lambda}{2} trace(w^T w) \\
&+ \sum_{r=1}^{n_r} L(r) H_1(w^T x_{r1} - w^T x_{r2}) \\
&+ \kappa \sum_{i=1}^{n} H_1\left(y_i(w^T x_i)\right) + H_1(-w_a) + H_1(-w_b) \\
&+ \left(1 - (w_a + w_b)\right)^2
\end{aligned}
\tag{3.6}
$$

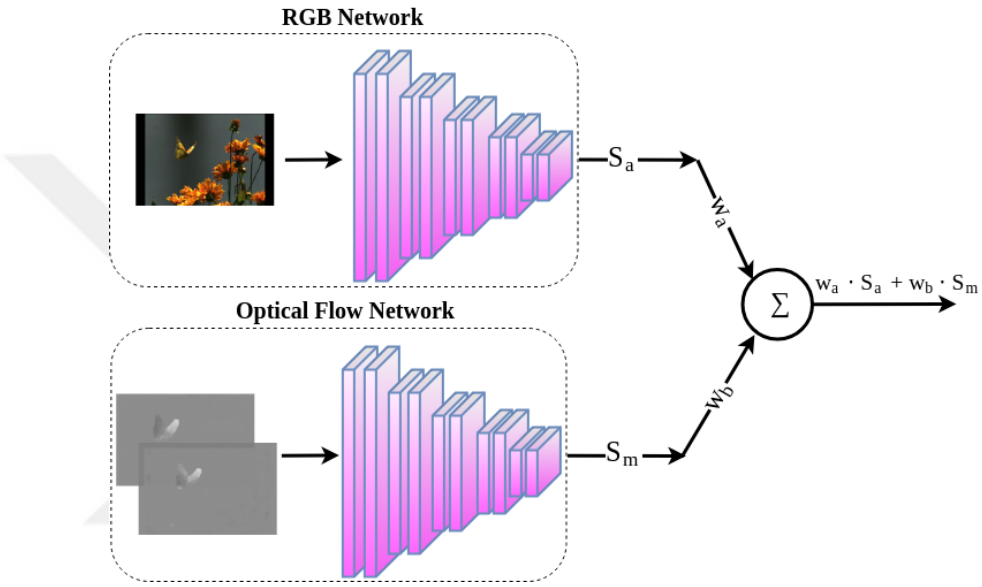


Figure 3.10. *The proposed two-stream architecture*

3.1.5. Implementation details

During the online tracking stage, first, a general search around the previous position of the target was applied followed by a local search. In general search, 256 candidate bounding boxes were drawn in each frame with varying aspect ratios and scales around the target location found in the previous frame. To this end, a ROI whose size and aspect ratio depended on the target's current bounding box was created. The bounding box of the target was at the center of the ROI, and the size of the ROI was about 5 times of the target bounding box size. Both random sampling and Edge Boxes [75], which is an effective region proposal method, were used to create candidate regions. The candidate region with the maximum classifier score in the general search was temporarily assigned as the new target location, then another local search was carried out around this location. In local search, 128 candidates were drawn around the temporary target location in an area smaller than the general searching area, and classifier scores were obtained using the two-stream

neural network. After determining the candidate regions with the highest scores in the general and local search, these scores were compared and the candidate region with the higher score was determined as the new location of the target object. This local search is very important for the proposed method, as the classifier of the network is very sensitive to even small target object position changes due to the ranking loss used in its optimization function. Also, if the final candidate region's score is greater than a given threshold by the user, this region is assigned as the new target position; otherwise, the new position is determined as a previous position. After determining the new location of the target object in the current frame, deep neural network-based two-stream networks should be updated. Due to speed issues, the weights of the *fc* layers (*fc4-6*) were updated, and the weights of the convolutional layers (*conv1-3*) were frozen in the network. To this end, 750 positives, 7000 negatives and 2000 sample pairs were drawn to fine-tune the pre-trained network in the first frame and in the other frames, 64 positives, 256 negatives, and 48 ranking pair samples, with a difference between *IoU* ratios larger than 0.4 were used. Since the ground-truth information of the target was given in the first frame, more samples were selected from this frame. To decide whether to update the model, a network model that was quite similar to the currently active model was cached, and this model was updated every 10 frames. The current active network model is updated if the score of the best target position is larger than the threshold value (threshold value = 0) pre-defined by the user. This proposed model update strategy differs from the strategy proposed in MDNet where short-term updates are performed only when potential tracking failures are observed. However, updating the model in each frame is crucial to track the scale and aspect ratio changes of the bounding boxes that frame the target object. In order to adapt the model to these changes, the maximum number of iterations was set to 10 during model updates.

In the cache, another model is needed to be kept because the active model was updated with each frame, rapidly adapting itself to smooth appearance changes. As a result, when the target undergoes an occlusion in a gradual way, the active model sometimes learns the background instead of the target object and cannot detect the object even if it appears again in the scene. To handle these problems, a cache model that ensures an update of that the active model was kept only when the target was mostly visible. Only one cache model was used; however multiple cache models can be used for different purposes based on different applications. For instance, consider aerial tracking of a non-

rigid object in the images captured by a UAV. Since both the target and the camera move, there would be various aerial target views in different orientations. When the target disappears from the scene due to occlusion and then reappears in the scene with a completely different view, the active model may find it difficult to detect the target object. In such cases, trained cache models with different views would help re-detect the target easily.

During offline training, 32 positive samples, 64 negative samples and 64 ranking pairs for 8 different frames (mini-batch size) were selected. Next, these selected patches were resized to $107x107$. The network was trained for 100 epoch on the ILSVRC 2015 dataset.

3.1.5.1. Architecture overview

Our one stream network architecture which is part of VGG-M network is shown in Table 3.5. The table shows structures, input and output shapes of the one-stream network architecture. In this table, *In* and *Out* represent a number of input and output channels of the filter, respectively, *K* denotes the filter size and *S* denotes the stride.

Table 3.5. *Architectures of the one-stream network*

Layer	Structure	Input Shape	Output Shape
Conv1	*In* = 3, *Out* = 96, *K* = 7, *S* = 2	$-1x3x107x107$	$-1x96x51x51$
MaxPooling	*K* = 3, *S* = 2	$-1x96x51x51$	$-1x96x25x25$
Conv2	*In* = 96, *Out* = 256, *K* = 5, *S* = 2	$-1x96x25x25$	$-1x256x11x11$
MaxPooling	*K* = 3, *S* = 2	$-1x256x11x11$	$-1x256x5x5$
Conv3	*In* = 256, *Out* = 512, *K* = 3, *S* = 1	$-1x256x5x5$	$-1x512x3x3$
Fc4	*In* = 4608, *Out* = 512	$-1x4608$	$-1x512$
Fc5	*In* = 512, *Out* = 512	$-1x512$	$-1x512$
Fc6	*In* = 512, *Out* = 1	$-1x512$	$-1x1$

3.1.5.2. Parameters of CNN

The input layer includes an image; thus, there are no parameters in input image. Consider a convolutional layer which receives input with *L* feature maps and has output feature map with *K* and has filter MxN size. The convolutional layer includes $(MxNxLxK)$ weights and *K* biases, so it has a total of $((MxNxL + 1)xK)$ parameters. Pooling layers just reduce the dimensions of inputs and it has no parameters. In the *fc*

layer, every neuron is connected to other neurons, and therefore, it has parameters. For C input neurons and P output layer neurons, it has PxC weights and C biases so the number of parameters here is $((P+1)xC)$). Table 3.6 shows the parameters of one-stream network with calculations for each layer.

3.1.5.3. Setting design parameters

There are 3 parameters to be determined by the user in the proposed method: the weight γ in $L(.)$, the weight κ of the proposed ranking loss that enforces the classifier to return positive scores for positive class samples and negative scores for negative class samples, as seen in Eq. (3.5) and the regularization parameter λ. To determine the best values of κ and γ, we randomly selected different videos from the OTB (Object tracking benchmark) [76], ImageNet [77] and VOT [40] datasets and evaluated accuracies for different values. The best accuracy was obtained for $\kappa = 0.5$ and $\gamma = 4.5$ values. Also, the regularization parameter (weight decay) λ was set to 0.0005.

Table 3.6. *Parameters of the one-stream network*

Layer (type)	**Output Shape**	**Calculations**	**Parameters #**
Conv2d-1	[-1, 96, 51, 51]	$(7x7x3+1)x96$	14,208
ReLU-2	[-1, 96, 51, 51]	-	0
LocalResponseNorm-3	[-1, 96, 51, 51]	-	0
MaxPool2d-4	[-1, 96, 25, 25]	-	0
Conv2d-5	[-1, 256, 11, 11]	$(5x5x96+1)x256$	614,656
ReLU-6	[-1, 256, 11, 11]	-	0
LocalResponseNorm-7	[-1, 256, 11, 11]	-	0
MaxPool2d-8	[-1, 256, 5, 5]	-	0
Conv2d-9	[-1, 512, 3, 3]	$(3x3x256+1)x512$	1,180,160
ReLU-10	[-1, 512, 3, 3]	-	0
Linear-11	[-1, 512]	$(4608+1)x512$	2,359,808
ReLU-12	[-1, 512]	-	0
Dropout-13	[-1, 512]	-	0
Linear-14	[-1, 512]	$(512+1)x512$	262,656
ReLU-15	[-1, 512]	-	0
Dropout-16	[-1, 512]	-	0
Linear-17	[-1, 1]	$(512+1)x1$	513
Total Parameters			4,432,001

3.2. Correlation Filter-Based Method

In the scope of the thesis, an object tracking algorithm that can run in real-time on the onboard computer on UAVs has been developed. To this end, firstly we proposed a hybrid method to track UAV captured by another UAV. Then, we generalized our proposed tracking method for general object tracking. To create this hybrid method, we employed KCF for its capability of fast-tracking, together with a relatively fast and accurate object detection method, YOLOv3. A new dataset of videos with various kinds of UAVs was created. This new dataset consisted of 15 UAV videos with 7500 frames in order to train YOLOv3 models and test our proposed method. Image 3.1 shows selected examples from our created new dataset. In addition to YOLOv3, YOLOV3-Tiny models were trained and used to achieve a faster hybrid tracking method.

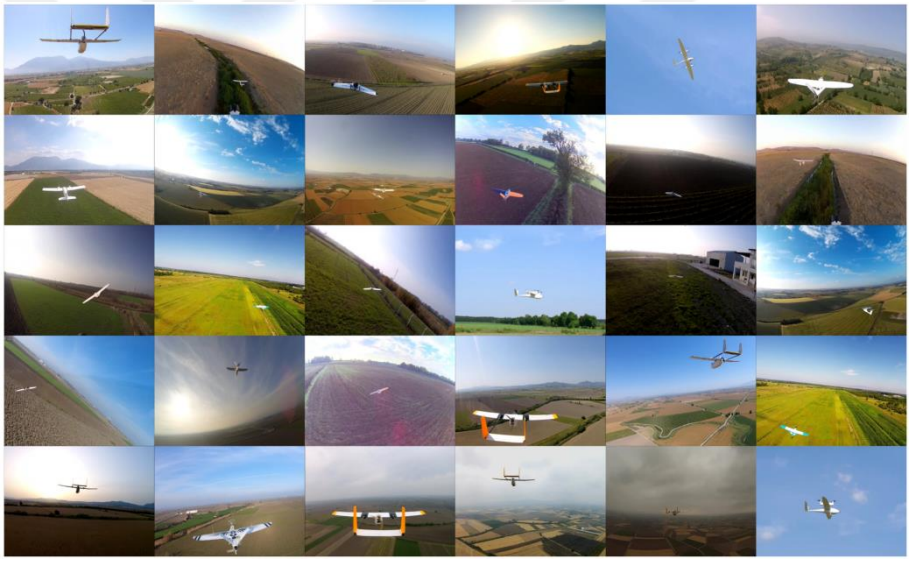


Image 3.1. *Samples from the UAV Tracking dataset created to train and test the UAV tracking methods*

3.2.1. Kernelized correlation filter tracker (KCF)

Correlation is a metric of similarity between two signals where signals of higher similarity are more highly correlated. In visual object tracking, correlation is used to measure the similarity between the two images. In visual target tracking algorithms, the correlation filter is designed to give a maximum response when applied to the target to be tracked. To speed up the algorithm and reduce the computational complexity of the learned filter, KCF [5] uses circulant matrices. KCF also uses HOG features [78] to improve tracking accuracy.

Now, let y_i represent a target and $f(z) = w^T z$ be the function that minimizes the squared error over samples x_i and their regression targets y_i. The KCF briefly solves the following ridge regression optimization problem,

$$\min_w \sum_i (f(x_i) - y_i)^2 + \lambda \|w\|^2 \tag{3.7}$$

where λ is a regularization parameter that controls over-fitting. The solution of the problem in the frequency domain corresponds to

$$\hat{w} = \frac{\hat{x}^* \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda} \tag{3.8}$$

where $\hat{w}$ indicates the Fourier transform of w, $\hat{x}^*$ is the complex conjugate of $\hat{x}$, and $\odot$ denotes the element-wise product. The w can easily be calculated by using the inverse Discrete Fourier Transform (DFT) in the spatial domain. In addition to the correlation filter, KCF employs the kernel trick to improve tracking accuracy.

Since the problem is solved in the frequency domain, the correlation filter is extremely fast, making it ideal for practical object tracking applications. As a result, KCF is one of the fastest methods in literature; it runs even on CPU in real-time, is robust against translation and scale differences and provides high accuracy in tracking. But when the target is occluded or out of the scope KCF easily drifts from the target since it does not have a self-correction mechanism and only searches in a ROI around the previous position target. In addition to these drawbacks, it cannot predict the aspect ratios of targets as in other correlation filter-based trackers, (To the best of our knowledge, there is only one study addressing this problem [1]).

3.2.2. Object detectors

Object detection methods can be roughly divided into two categories as one-stage and two-stage detectors. Two-stage networks firstly propose regions using region proposal networks and then apply classification and regression on these proposals. Figure 3.11 shows Faster R-CNN [79] region proposal network (RPN) which is the well-known two-stage network.

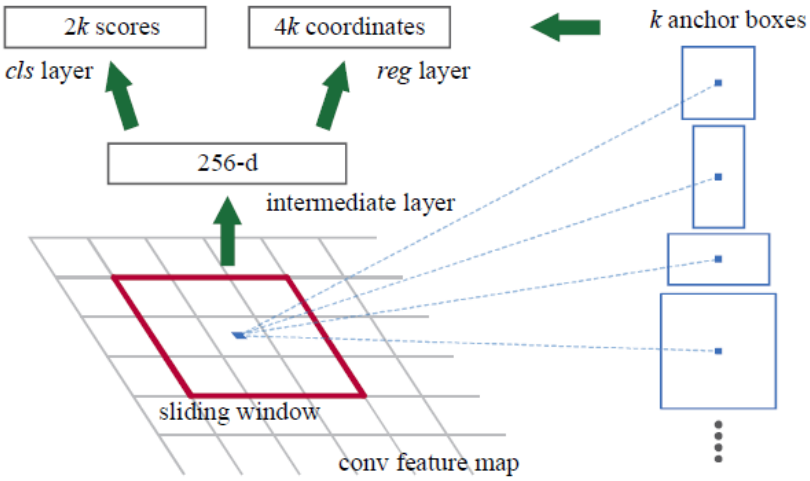


Figure 3.11. *Faster R-CNN - Region proposal network [79]*

The other one-stage methods skip region proposal stage and directly predict bounding boxes and their class probabilities using a one-stage network. Redmon et al. [4] proposed a one-stage end-to-end learnable object detector network called YOLO. Figure 3.12 shows that YOLO architecture. YOLO is extremely faster than two-stage detectors because it divides the input image into *S x S* grid before it feeds the convolutional neural network with the whole this image, but two-stage detectors firstly determine candidate region proposal using RPN and non-maximum suppression (NMS), then they predict bounding boxes and class probabilities. YOLO determines class probabilities and bounding boxes of objects using following loss function.

$$\begin{aligned}
&\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2] \\
&\quad+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right] \\
&\quad+\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}\left(C_i-\hat{C}_i\right)^2 \\
&\quad+\sum_{i=0}^{S^2}1_{i}^{obj}\sum_{c\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2
\end{aligned} \tag{3.9}$$

YOLO is a fast object detection method, not an object tracking method. It cannot distinguish the target object in a scene from other objects in the same class. Using YOLO in video streams in tracking objects can cause discontinuity, as it is not designed to return

a detection response for each frame. Power consumption and run-time are extremely crucial with UAV-specific vision applications. While YOLO runs relatively fast on GPUs compared to two-stage detectors, it cannot be compared to KCF which runs with 100+ FPS on a CPU. Also, the YOLOv3-Tiny model which has a smaller number of convolutional layers is used to improve run-time performance.

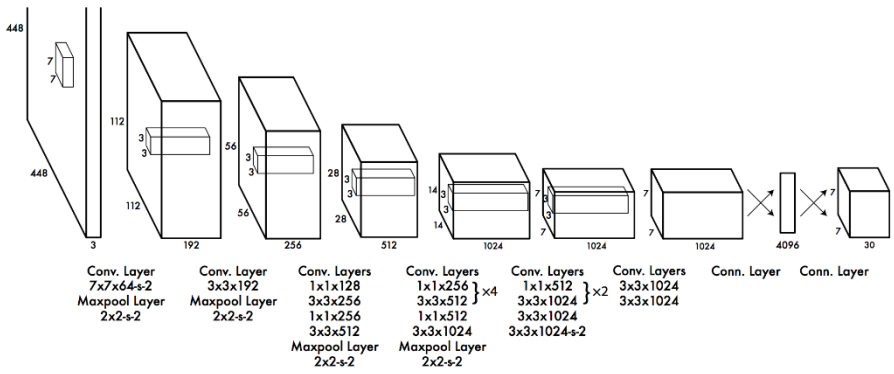


Figure 3.12. *YOLO architecture [4]*

3.2.3. The proposed hybrid tracker-UAVH

YOLO is a fast general object detector with high accuracy. In a single object tracking, targets and backgrounds change at each video stream, so object tracking is class agnostic. It also runs at ~15 FPS on onboard computers. KCF is an extremely fast tracker, but the target is not robust to changes in the aspect ratios of the bounding boxes and this causes it to easily drift from the target. Given the advantages and limitations of these two methods, they seem to complement each other. Therefore, a hybrid tracker is proposed to take advantage of these two methods. The proposed hybrid tracker UAVH uses YOLO to detect the target in the first frame and re-detect it when the tracker fails and, employs KCF to keep tracking the target object. In this way, a new hybrid tracker was created using both the YOLO detector and the KCF tracker.

To perform the proposed method, a YOLO model was needed to be trained first to detect objects. To this end, YOLO models were trained separately to detect UAVs and general objects using the created dataset and MS-COCO dataset. For the best run-time performance, the original C implementation was compiled as a dynamic library instead of the python implementation of the YOLO method and was used in our Python code by calling these functions. After YOLO had been trained for specific categories to be used

for tracking, a mechanism was needed to switch between detector and tracker approaches. The proposed method relies on tracker scores: If the KCF tracker's filter score is less than a given threshold value, the YOLO detector is called to detect all UAVs (or any other tracked object sample) in the frame. Next, the proposed method relies on YOLO scores to decide whether the object of interest is in the frame. The scores of the regions where YOLO returns are checked and if the scores of these regions are higher than the given thresholds, they are determined as candidate locations.

The *IoU* ratios are calculated between these k candidate regions (P_{D_i}) and the tracker's latest output (P_T) using equation (3.10). If there is at least one candidate region with an *IoU* ratio greater than zero, the candidate with the maximum *IoU* score is determined as a new position of the tracked target and it is utilized to re-train KCF.

$$\max_{i \in k} \left(\frac{|P_T \cap P_{Di}|}{|P_T \cup P_{Di}|} \right) \tag{3.10}$$

If there is no intersection between the candidates and the latest position of the object, that is, the target moves away, the Euclidean distances are calculated between the centers of the candidate boxes (x_{D_i}, y_{D_i}) and the latest output of the tracker (x_T, y_T) using equation (3.11). Finally, the closest candidate position is determined as the location of the target.

$$\max_{i \in k} (\sqrt{(x_T - x_{Di})^2 + (y_T - y_{Di})^2}) \tag{3.11}$$

By following the above procedure, the tracker is enabled to recover from a failure and update the aspect ratio dynamically. Since the YOLO runs only when the tracker's filter score is less than a given threshold, the proposed method runs faster than YOLO. As a result, the proposed method used KCF to track the single object and got continuous results and became faster, while using YOLO to recover and dynamically update the aspect ratio when the tracker fails.

4. EXPERIMENTS

We conducted two different experiments on different datasets to evaluate our deep learning-based (Ranking) and correlation filter-based (UAVH) approaches. First, we tested the Ranking method we had developed for general single object tracking, and then the UAVH method we had developed for UAV tracking applications using onboard computers.

4.1. Deep Learning-Based Tracker Experiments

We tested our deep learning-based methods, RankingAF and RankingSF on 8 different single object tracking datasets and evaluated their results. These well-known datasets are the OTB-2015 [76], TColor-128 (Temple-Color) [80], UAV123 (Unmanned Aerial Vehicle) [81], DTB70 (Drone Tracking Benchmark) [82], NfS (Need for Speed) [83], LaSOT (Large-Scale Single Object Tracking) [84], VOT-2017/2018 (Visual Object Tracking) [40], GOT-10k (Generic Object Tracking Benchmark) [85]. RankingAF directly adds the scores of RGB and Optical networks while RankingSF learns network-specific weights as described in the method. The proposed approaches were implemented both in Python using Pytorch and in Matlab using matconvnet and ran on a Quadro P5000 16GB GPU and an eight-core 2.1 GHz Intel Xeon E5-2600 processor. To pre-train our two-stream network, we used ILSVRC 2015 datasets including 2156 video streams, except the GOT-10k dataset where we utilized its train split for a fair comparison. On a single Nvidia Quadro P5000 GPU, we achieved a tracking speed of 3 and 4 FPS in Matlab and Python, respectively. We compared our methods to recently published state-of-the-art methods (Table 4.1).

4.1.1. Evaluation protocols

The results of the trackers are reported using the one-pass evaluation (OPE) protocol with both success and precision scores. To generate the performance scores and the plots, we used the OTB evaluation toolkit on the OTB-2015, UAV123, Temple-Color-128, and DTB70 datasets. The precision plot, which is a widely used evaluation metric on tracking, shows the percentage of frames whose center of predicted locations are within the given threshold (we used 20 pixels for our experiments) of the center of ground-truths. These center location errors are calculated using Euclidean distance. The success plot employs the bounding box overlapping between predicted target bounding boxes and ground-truth to show performances of trackers and shows the ratios of

successful frames at the thresholds ranging from 0 to 1. Finally, trackers are ranked using the area under the curve (AUC) obtained from these plots instead of using a given threshold. We used AUC scores to compute the performances of the trackers on the NfS dataset. On the GOT-10k datasets, we used normalized precision (Pre_{norm}) in addition to AUC and precision (Pre) scores. We used average overlap (AO) and success rates (SR) at overlap thresholds set to 0.5 and 0.75 on the LaSOT benchmark. Finally, we used the VOT toolkit evaluation protocol to evaluate the performances of the trackers on the VOT-2017/2018 datasets. This protocol applies a reset-based methodology in which a tracker re-initialized whenever tracking fails using new ground-truth information and also it uses no-reset based methodology using average overlap (AO) metric as in OTB protocol. In the reset-based methodology, the performance is determined in terms of expected average overlap (EAO) which is obtained using both Accuracy (A) and the number of re-initialization (robustness-R).

Table 4.1. *State-of-the-art tracker list*

Tracker name	Publisher	Year of publication
TADT [86]	CVPR	2019
GCT [69]	CVPR	2019
VITAL [87]	CVPR	2018
CFCF [38]	T-IP	2018
SiamRPN [50]	CVPR	2018
FlowTrack [67]	CVPR	2018
LSART [88]	CVPR	2018
ECO [24]	CVPR	2017
MCPF [89]	CVPR	2017
SiamFCv2 [36]	CVPR	2017
BACF [90]	ICCV	2017
CFWCR [91]	ICCVW	2017
CSRDCF [92]	CVPR	2017
GNet [93]	ICIP	2017
MDNet [3]	CVPR	2016
CCOT [39]	ECCV	2016
GOTURN [11]	ECCV	2016
DeepSRDCF [35]	ICCV	2015
CF2 [34]	arXiv	2015

4.1.2. State-of-the-art comparison

4.1.2.1. Object tracking benchmark (OTB)

The OTB [76] is a well-known single object tracking dataset. We report the results of our methods RankingAF and RankingSF on the OTB-2015 which includes 100 video sequences (590 frames average length), with 11 different challenging factors. Figure 4.1 shows the precision and success plots of the recently published state-of-the-art methods including ECO, VITAL, MDNet , CFCF, CCOT, TADT, DeepSRDCF, and CF2 on the OTB-2015 dataset. In addition to these plots, in Table 4.2, we compare AUC and precision scores of our methods with other state-of-the-art methods. The proposed RankingSF and RankingAF had the best precision scores of 92.1% and 92.0% respectively. In terms of AUC scores, RankingSF and RankingAF achieved a 68.8% and 68.3% AUC, respectively, which are close to the best result of ECO achieved by 69.1%.

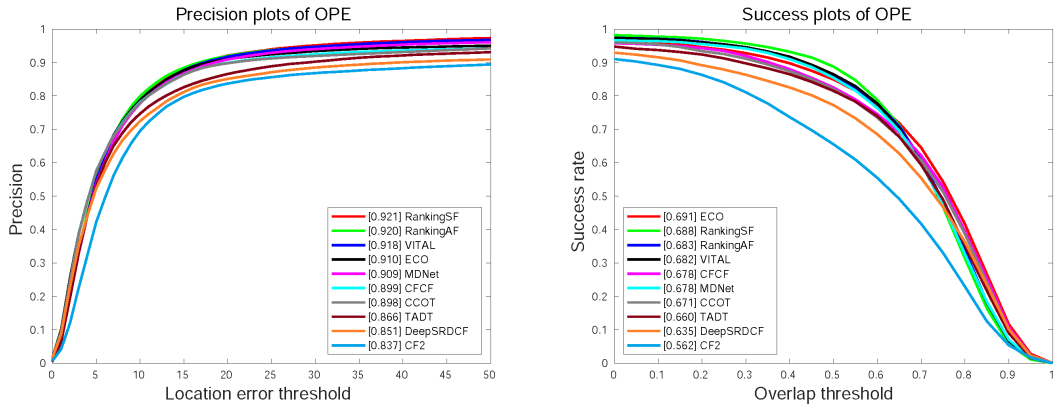


Figure 4.1. *Precision and success plots on the OTB-2015 dataset using the OPE protocol*

The AUC and precision scores are also reported in Figure 4.2 and Figure 4.3, respectively for 11 different attributes in the OTB-2015 dataset, including fast motion, background clutter, motion blur, deformation, illumination variation, in-plane rotation, low resolution, occlusion, out-of-plane rotation, out of view and scale variation.

In terms of AUC, our trackers obtained either the best or comparable results in these challenging attributes. By using motion clues through the optical flow network, the AUC scores were improved, and the proposed RankingSF achieved the best AUC except for occlusion, motion blur and illumination variation challenges. For the motion blur challenge, the motion stream (optical flow network) yields insensitive scores due to

motion blur frames, so reducing the performance of our trackers. Thanks to the score fusion mechanism, RankingSF tolerates insensible scores of the motion network better than RankingAF. Similarly, for the occlusion challenge, the motion stream cannot produce any sensible score as there is no movement for long time periods in the video sequences. As a result, the AUC of the proposed methods slightly drop. Unlike the occlusion and motion blur challenges, our trackers achieve comparable AUC scores in the illumination variation challenge.

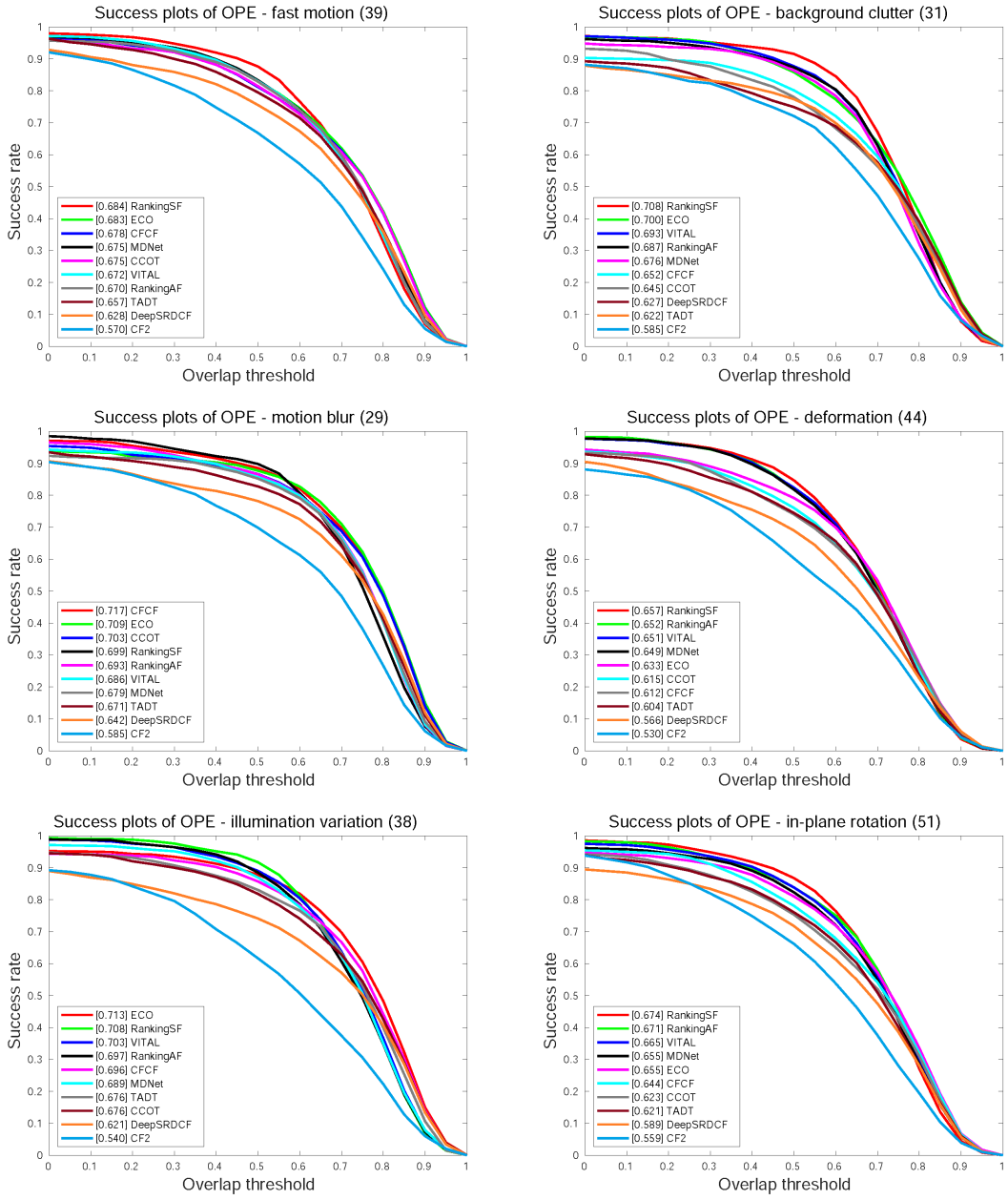


Figure 4.2. *Comparison of the AUC scores for 11 different attributes on the OTB-2015 dataset*

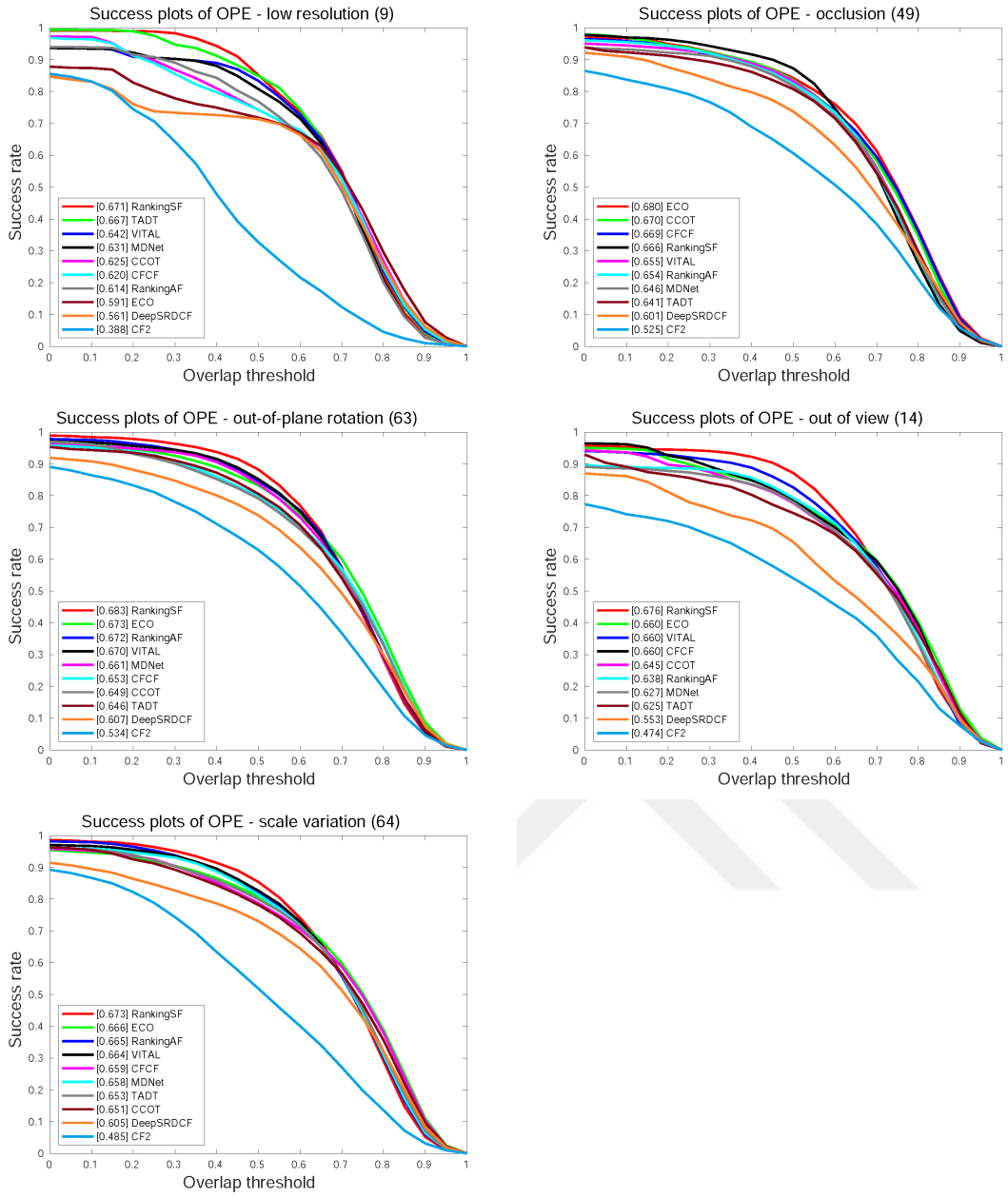


Figure 4.2. (continued) *Comparison of the AUC scores for 11 different attributes on the OTB-2015 dataset*

In terms of precision plots, our trackers achieved the best or comparable results in these 11 different attributes. Using motion information through the optical flow network improved the precision scores; the proposed RankingSF especially achieved the best precision scores in 7 different attributes except for occlusion, out of view, low resolution, and deformation challenges. In the deformation challenge, RankingAF achieved the best precision score followed by our other tracker RankingSF. Similar to AUC scores, RankingSF tolerated insensible scores of the motion network better than RankingAF

thanks to the score fusion mechanism. For the occlusion challenge, the motion stream cannot produce any sensible score as there was no movement for long time periods in the video streams as in the success plot. For low resolution, RankingSF achieved a 99.7% precision score which was very close to the best result of 99.9% obtained with TADT.

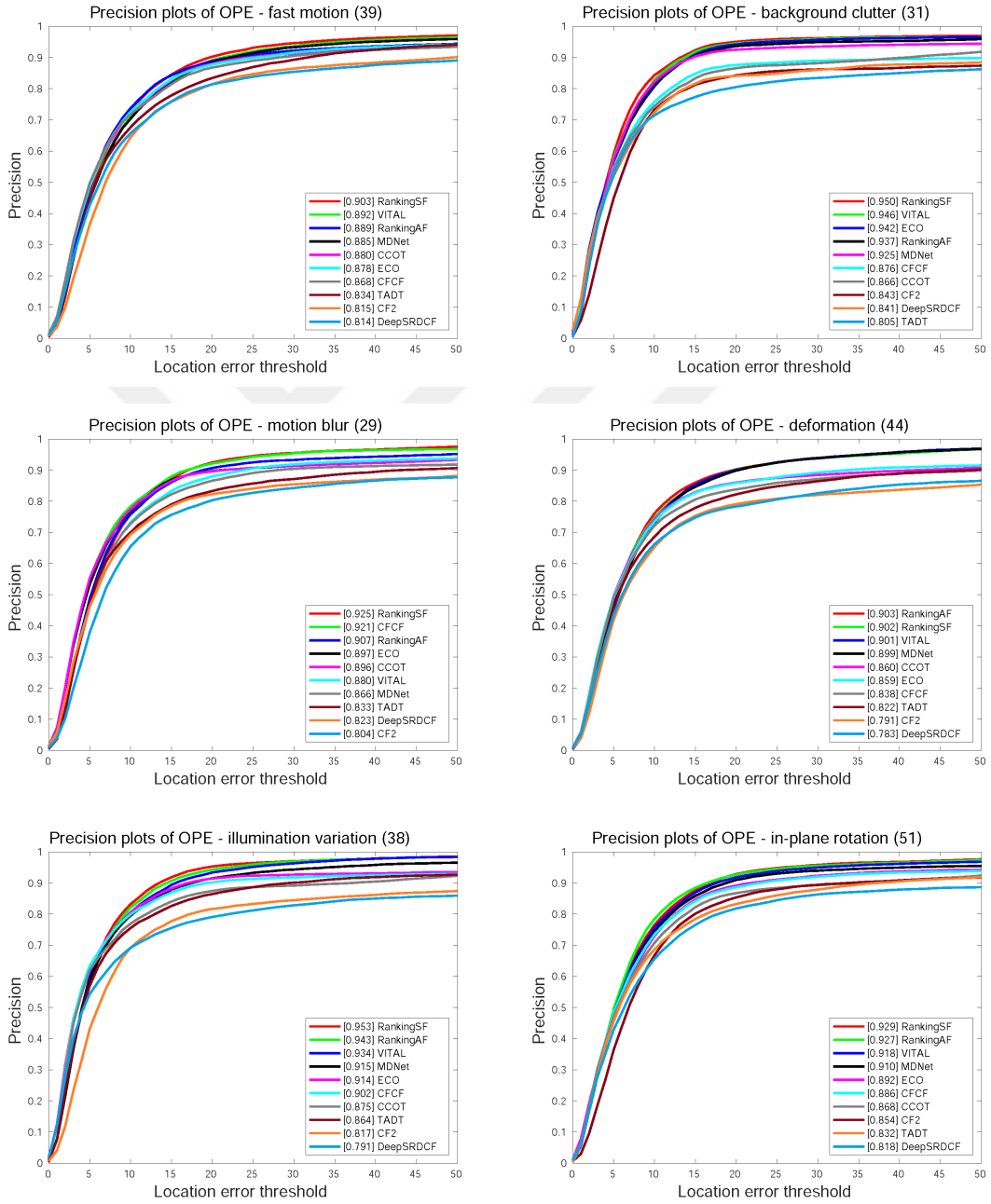


Figure 4.3. *Comparison of the precision scores for 11 different attributes on the OTB-2015 dataset*

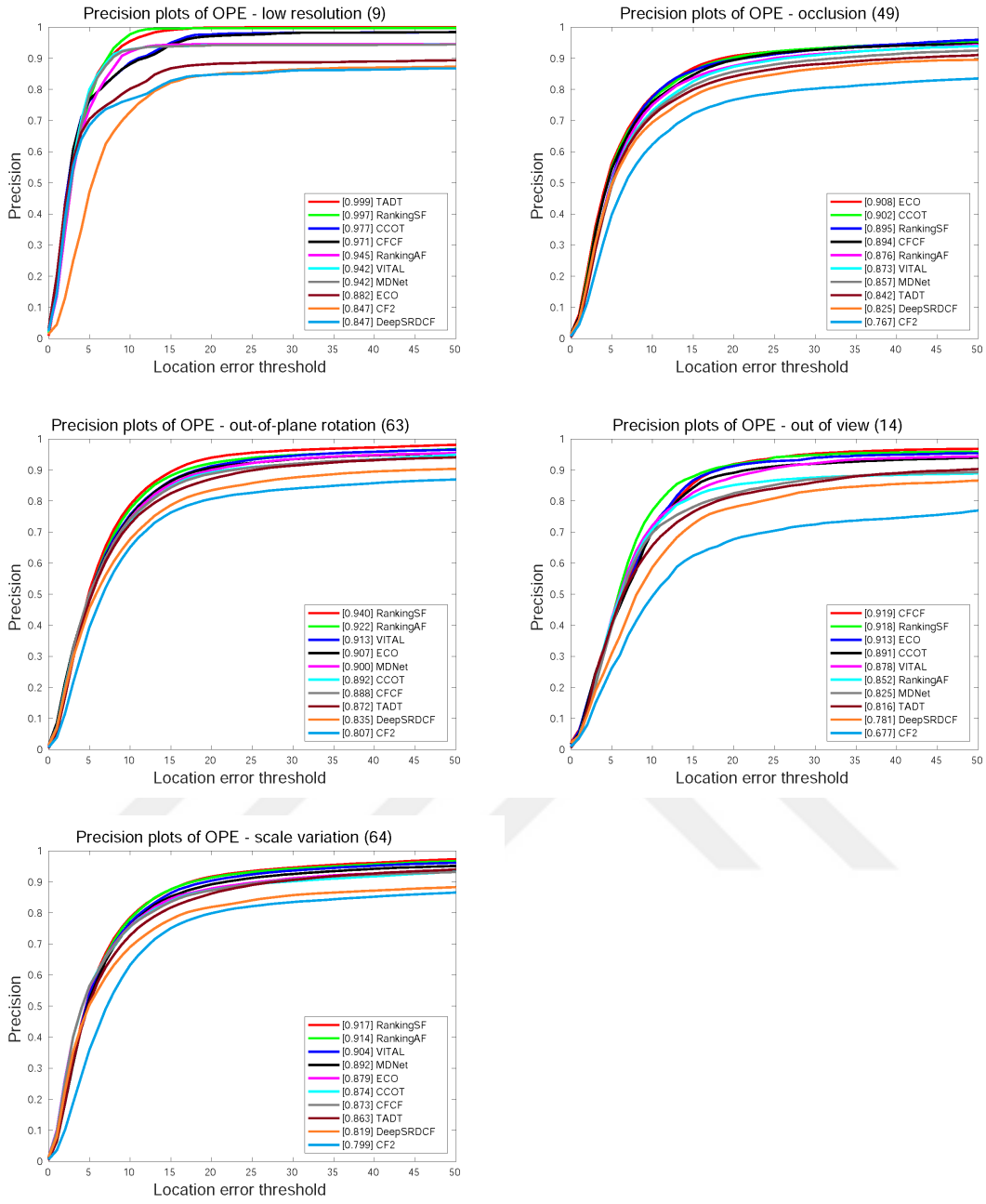


Figure 4.3. (continued) *Comparison of the precision scores for 11 different attributes on the OTB-2015 dataset*

4.1.2.2. Temple color 128 (TColor-128) dataset

The TColor-128 [80] contained 128 color-based videos, including 50 videos used in the previous benchmarks and 78 videos that had never been used before for object tracking. This benchmark was also labeled with 11 challenging factors as in OTB-2015 dataset. Table 4.2 shows the AUC and precision scores of our trackers and the recently published state-of-the-art methods including ECO, VITAL, MDNet, TADT, and GCT on

the TColor-128 benchmark. The proposed RankingSF and RankingAF had the best precision scores of 82.0% and 81.5%, respectively as in the OTB-2015 precision. In terms of AUC score, the proposed RankingAF obtained comparable results on this dataset.

Table 4.2. *State-of-the-art comparison on the OTB-2015 and TColor-128 datasets in terms of AUC and precision scores. (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	OTB-2015		TColor-128	
	AUC	**Precision**	**AUC**	**Precision**
RankingAF	68.3	92.0	57.6	81.5
RankingSF	68.8	92.1	55.6	82.0
MDNet	67.8	90.9	56.3	77.7
ECO	69.1	91.0	59.7	80.0
VITAL	68.2	91.8	59.7	80.0
SiamRPN	63.6	85.0	56.3	77.7
GCT	64.8	85.4	59.7	80.0
TADT	66.0	86.6	56.2	-
FlowTrack	65.5	88.1	-	-

4.1.2.3. Unmanned aerial vehicle 123 (UAV123) dataset

The UAV123 [81] contained 123 low-altitude aerial video sequences captured by an UAV. Table 4.3 shows the AUC and precision scores of our proposed and the recently published state-of-the-art methods including ECO, VITAL, MDNet, SiamRPN and UAVH on the UAV123 benchmark. We report the precision and success scores of the tested methods in Table 4.2. The proposed RankingAF and RankingSF had the best performances of 53.6% and 53.5%, respectively in terms of AUC scores. In terms of precision, while RankingSF achieved the third-best result, RankingAF obtained competitive results.

4.1.2.4. Drone tracking benchmark 70 (DTB70)

Similar to the UAV123 dataset containing aerial videos, this benchmark [82] contained 70 high-diversity videos shot by drone cameras. We compare our trackers RankingAF and RankingSF with our baseline method MDNet in Table 4.3. Our proposed RankingAF and RankingSF trackers obtained the best and the second-best scores, respectively. Notably, the proposed RankingAF significantly outperformed our baseline MDNet with relative gains of 6.95% and 8.11%, respectively in terms of precision and AUC.

Table 4.3. *State-of-the-art comparison on the UAV123 and DTB70 datasets in terms of AUC and precision scores (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	UAV123		DTB70	
	AUC	**Precision**	**AUC**	**Precision**
RankingAF	53.6	74.2	49.3	73.8
RankingSF	53.5	74.6	47.5	71.9
MDNet	52.8	74.7	45.6	69.0
ECO	52.5	74.1	-	-
Vital	52.5	74.1	-	-
SiamRPN	52.7	74.8	-	-
UAVH	52.7	74.3	-	-

4.1.2.5. Generic object tracking benchmark (GOT-10k)

The GOT-10k [85] is a large-scale object tracking benchmark that includes more than 10,000 videos containing 563 object classes and 87 motion classes, and more than 1.5 million manually labeled high precision bounding boxes. Using the GOT-10k protocol, we evaluated our trackers on the GOT-10k test split which contains 420 videos and 83 classes and reported the results of the proposed methods and other state-of-the-art methods including MDNet, ECO, GOTURN and SiamFCv2 in Table 4.4. We pre-trained our trackers only on the GOT-10k train split for a fair comparison. In terms of average overlap (AO) and success rates (SR) at overlap thresholds set to 0.5 and 0.75, the proposed RankingSF obtained the best and second-best scores, respectively. In addition, the other proposed method, RankingAF, obtained the third-best scores in terms of AO and $SR_{0.5}$.

Table 4.4. *State-of-the-art comparison on the GOT-10k test set in terms of average overlap (AO), success rates (SR) at overlap thresholds 0.5 and 0.75 (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	$SR_{0.5}$(%)	**$SR_{0.75}$(%)**	**AO(%)**
RankingAF	37.5	10.9	36.1
RankingSF	39.7	11.2	37.8
MDNet	30.3	9.9	29.9
ECO	30.9	11.1	31.6
GOTURN	37.5	9.8	34.7
SiamFCv2	40.4	14.4	37.4

4.1.2.6. Need for speed (NfS) dataset

The NfS [83] dataset is the first higher frame rate (30 and 240 FPS) benchmark for real-world visual object tracking. It contains 33 classes and100 manually labeled video streams with an average frame length of 3830 frames. We tested our methods on 240 FPS version of the NfS. Table 4.5 shows the results of the proposed and state-of-the-art trackers including MDNet, SiamFC, BACF and GOTURN. Our methods, RankingAF and RankingSF, achieved the best and second-best scores in terms of AUC, respectively. As seen in table, the proposed trackers also significantly outperformed our baseline method MDNet tracker by about 10% in AUC.

Table 4.5. *State-of-the-art comparison on the NfS (240 FPS version) dataset. (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	RankingAF	**RankingSF**	**MDNet**	**SiamFC**	**BACF**	**GOTURN**
AUC (%)	56.6	56.2	47.3	52.0	49.5	37.6

4.1.2.7. Large-scale single object tracking (LaSOT) dataset

The LaSOT [84] dataset includes 1400 video streams with 3.52 million high-quality fully annotated bounding boxes. We evaluated our methods on the test split which includes 70 classes and 280 videos. Results of our methods and other methods including MDNet, ECO, VITAL, BACF and SiamFC on the LaSOT are reported in Table 4.6 in terms of AUC, precision (Pre) and normalized precision (Pre_{norm}). The proposed RankingSF tracker achieved the second-best precision and third-best AUC and normalized precision scores. Nonetheless, the proposed methods achieved comparable results with MDNet, a top-performing tracker.

Table 4.6. *State-of-the-art comparison on the LaSOT dataset. (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	AUC (%)	**Pre (%)**	**Pre_{norm} (%)**
RankingAF	38.2	35.2	42.1
RankingSF	38.4	36.6	43.8
MDNet	39.7	37.3	46.0
ECO	32.4	30.1	33.8
VITAL	39.0	36.0	45.3
BACF	25.9	23.9	28.3
SiamFC	33.6	33.9	42.0

4.1.2.8. Visual object tracking (VOT) dataset

The VOT-2017 [40] dataset includes 60 video streams selected from about 390 videos, with 6 different attributes including illumination change, camera motion, motion change, occlusion and size change. The VOT-2017 and 2018 have the same videos in the short-term challenge in which we tested the proposed RankingAF and RankingSF trackers. We compared our methods with other state-of-the-art trackers including LSART, CFCF, ECO, CCOT, CFWCR, GNet, CSRDCF, and MCPF. The results of the trackers are reported in Table 4.7 in terms of accuracy (A), robustness (R), expected average overlap (EAO) and average overlap (AO) by using the VOT evaluation toolkit. Except in terms of robustness, our methods RankingAF and RankingSF obtained the best and second-best scores, respectively. Especially in terms of Accuracy and AO, our methods significantly outperformed other trackers including CFCF which is VOT-2017 short-term challenge winner. However, robustness performances of LSART and GNet were better than proposed trackers.

Table 4.7. *State-of-the-art comparison on the VOT-2017/2018 benchmark (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	EAO	Accuracy	Robustness	AO
RankingAF	0.32	0.52	16.23	0.50
RankingSF	0.31	0.51	17.41	0.49
LSART	0.32	0.48	11.20	0.44
GNet	0.27	0.49	13.33	0.42
ECO	0.28	0.46	15.33	0.40
MCPF	0.25	0.51	21.29	0.44
CFCF	0.29	0.49	15.17	0.38
CCOT	0.27	0.46	17.67	0.39
CFWCR	0.30	0.48	14.33	0.37
CSRDCF	0.26	0.49	18.50	0.34

We also evaluated results for 6 different challenging attributes. As seen in Table 4.8, our proposed trackers, RankingAF and RankingSF, achieved the best and second-best scores, respectively, for illumination change, motion change, and size change challenges in terms of accuracy. For occlusion challenge, our methods once again achieved the best result but RankingSF was better than RankingAF for this challenge. Moreover, RankingAF and RankingSF significantly outperformed other top-performing

trackers for both accuracy and robustness on motion and size changes challenges. The proposed trackers obtained the best scores for size changes since the proposed trackers better adapted to scale/aspect ratio changes compared to other state-of-the-art trackers thanks to ranking loss (proposed term) in our methods. Obtaining both the highest accuracy and robustness performances for motion change challenge were expected since our motion network used motion cues for object tracking. Our methods only failed to outperform other state-of-the-art trackers on camera motion since the whole scene moves with the camera in this challenge, and the motion network in the proposed methods does not discriminate the object movement from the background.

Table 4.8. *State-of-the-art comparison on the VOT-2017/2018 dataset in terms of accuracy and robustness scores for different challenging attributes (The red, blue and green fonts indicate the top-performing trackers, respectively)*

		RankingAF	RankingSF	LSART	ECO	CFWCR	CFCF	GNet	CCOT
Camera Motion	**A**	0.52	0.51	0.51	0.51	0.54	0.54	0.55	0.52
	R	26.00	28.00	19.73	25.0	28.00	29.00	23.00	26.00
Illum. Change	**A**	0.52	0.51	0.48	0.50	0.48	0.44	0.50	0.45
	R	1.00	3.00	0.53	4.00	3.00	0.00	1.00	6.00
Motion Change	**A**	0.57	0.54	0.50	0.48	0.49	0.51	0.50	0.48
	R	7.67	8.00	8.67	18.0	12.00	15.00	14.00	19.00
Occlusion Challenge	**A**	0.51	0.52	0.44	0.35	0.39	0.44	0.41	0.39
	R	19.00	20.00	20.80	22.0	19.00	20.00	14.00	22.00
Size Change	**A**	0.55	0.52	0.46	0.45	0.45	0.49	0.49	0.45
	R	5.00	6.00	9.00	9.00	12.00	8.00	9.00	14.00

4.1.3. Ablation studies

To show the importance of the ranking loss term, hard ranking mining and two-stream network with different fuse strategies, we performed an ablation study on four different datasets including OTB-2015, TColor-128, DTB70, and VOT-2017/2018. Table 4.9 shows the components used by each method. As seen in the table, RankingT-NR uses only hinge loss and single-stream neural network (RGB network) to demonstrate the effect of the ranking loss term. We added this term with randomly chosen ranking pair samples to RankingT-NR which uses the single-stream network as in RankingT-NR. Next, we added hard sample mining strategy to RankingT tracker for ranking loss. To this end, instead of choosing the random sample pairs, xr1 and xr2, for ranking loss term, we first sorted the scores of xr2 (the visual features of the region that frames the object with

less accuracy). Then, we paired xr1 samples with xr2 samples having the highest confidence scores and we named this tracker as RankingT-HM. To show the importance of the two-stream network, we used both spatial and motion networks with different fusion strategies. The only difference of these trackers from RankingT-HM is that they use two-stream networks instead of a single-stream. These methods are called RankingAF and RankingSF and we used these trackers to compare with state-of-the-art trackers. RankingSF learns network-specific weights while RankingAF directly adds the scores of RGB and Optical networks as described in the method. All of these trackers are pre-trained on the ILSVRC 2015 dataset, so they are directly comparable.

Table 4.9. *Usage of the proposed components by trackers*

	Hinge Loss	Ranking Loss	Hard-Mining	Single-Stream	Two-Stream
RankingT-NR	✓	✖	✖	✓	✖
RankingT	✓	✓	✖	✓	✖
RankingT-HM	✓	✓	✓	✓	✖
RankingAF	✓	✓	✓	✖	✓
RankingSF	✓	✓	✓	✖	✓

Table 4.10 and Table 4.11 show the results of the described trackers on different datasets. It can be seen from the tables that adding ranking loss term improved all scores on all of the 4 different datasets. More precisely, it improved both AUC and precision scores of around 2% on the OTB-2015 dataset. On the TColor-128 and DTB70 datasets, similarly, both AUC and precision scores were improved by around 2%. Especially, on the VOT2017/2018 dataset, RankingT achieved an EAO score of 0.27, with a relative gain of 7% over without ranking loss tracker, RankingT-NR.

Next, we investigated the impact of the hard sample mining strategy for the proposed ranking loss term. The given results in the tables indicate that adding this component increased all scores on all datasets. Especially, on the VOT2017/2018 dataset, RankingT-HM significantly outperformed RankingT method by 2% in terms of both A and EAO.

Lastly, we investigated the impact of using two-stream network by optical flow in addition to RGB features. On the OTB-2015 and TColor-128 datasets, results of RankingT-HM and two-stream methods were similar, because these datasets mostly had

the same videos. On the DTB70 dataset, RankingAF and RankingSF obtained the best results, respectively. Notably, RankingAF achieved an AUC of 49.3% and 73.8% which were around 2% higher than the results of the RankingT-HM. For RankingAF, using the two-stream network led to a significant improvement on the VOT-2017/2018 dataset, giving an EAO of 0.32, with a relative gain of almost 10% over RankingT-HM.

Table 4.10. *Ablation studies on the OTB-2015 and TColor-128 datasets (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	OTB-2015		TColor-128	
	AUC	**Precision**	**AUC**	**Precision**
RankingT	68.3	91.2	57.6	81.5
RankingT-NR	66.4	89.4	56.4	80.3
RankingT-HM	68.9	91.6	58.2	82.0
RankingAF	68.3	92.0	57.6	81.5
RankingSF	68.8	92.1	58.3	82.1

These ablation studies show that usage of the ranking loss and hard-sample mining is crucial for object tracking. Moreover, the two-stream network using motion information in addition to the appearance information obtains better accuracies compared to the tracking network using appearance information alone.

Table 4.11. *Ablation studies on the DTB70 and VOT-2017/2018 datasets. (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	DTB70		VOT-2017/2018		
	AUC	**Precision**	**EAO**	**Accuracy**	**Robustness**
RankingT	45.9	71.0	0.27	0.50	19.84
RankingT-NR	44.7	69.6	0.25	0.47	22.64
RankingT-HM	47.3	71.9	0.29	0.52	18.02
RankingAF	49.3	73.8	0.32	0.50	16.23
RankingSF	47.5	71.9	0.31	0.51	17.41

We updated the w_a and w_m values during online tracking. Figure 4.4 shows the computed values of w_a and w_m along with a complete video stream for zebrafish. Magenta dashed line shows the frames with occlusion in the given frame and green dashed line

shows an abrupt motion in the given frame. Red and blue dashed lines show w_a and w_m weights, respectively, in the given frame. As shown in the given figure, the proposed two-stream network with score fusion strategy generally relies on the spatial network. But, the weight of the temporal network dominates especially when there are some abrupt motion changes as expected (not all the time, among 35 frames with abrupt motion changes, the temporal network dominates for 30 frames). For this zebrafish video stream, the weight for the temporal network is also higher in the occluded frames (frame no: 248-251). This shows that the score fusion strategy works as expected and the system learns to give higher weights to the temporal network when the spatial network fails.

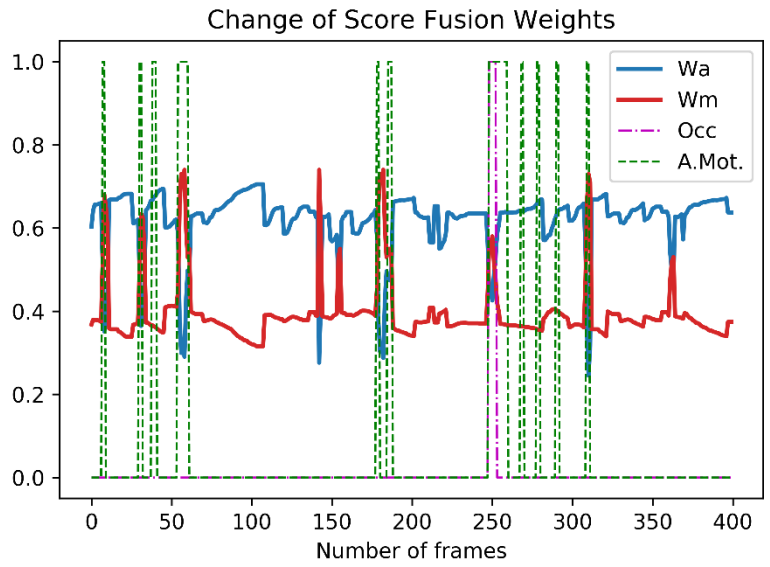


Figure 4.4. *Change of score fusion weights on zebrafish video sequences during online tracking*

4.1.4. Visual comparison

Finally, we are showing a comparison of the proposed method, RankingAF, with state-of-the-art correlation filter tracker ECO and deep neural network tracker (MDNet) on two challenging video sequences (fish, diving). Aspect ratios of bounding boxes and object appearances that frame targets dramatically change in these video sequences. In the given Image 4.1, the first and the last images correspond to the second and the last frames of the videos, while the other images correspond to consecutive intermediate frames. In the first video frames (fish), while our method achieved to track the target until the end of the video frames, both ECO and MDNet failed to track the target object. In the second video sequence (diving), all of the methods achieved to track the target object, but our tracker, RankingAF, returned better bounding boxes.

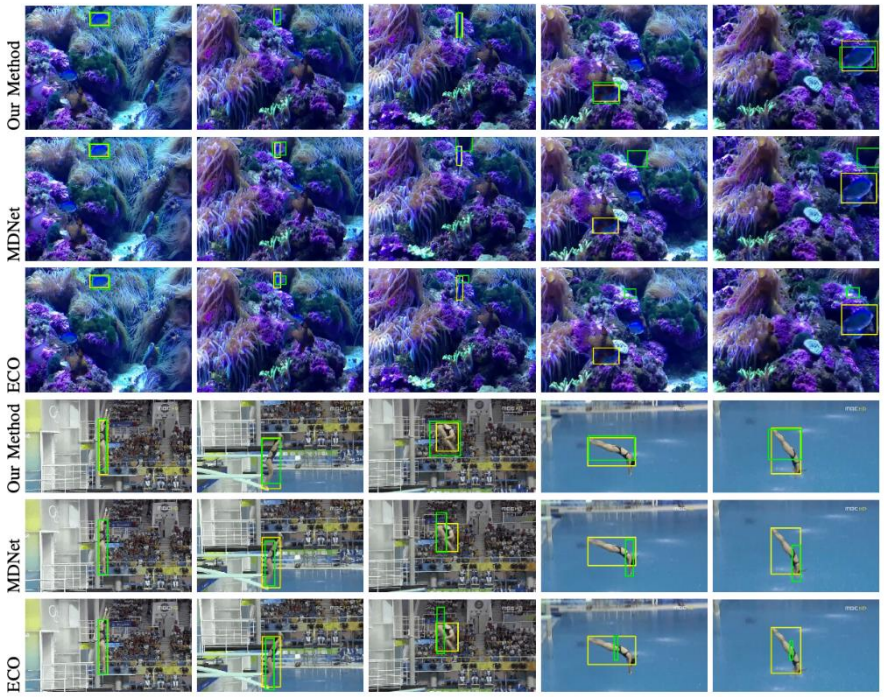


Image 4.1. *A comparison of the proposed method, RankingAF, with MDNet and ECO.*

In Image 4.1, the green bounding boxes show results of the trackers and the yellow bounding boxes show the ground-truth. The proposed two-stream method returned more precise bounding boxes framing the target. As a matter of fact, the returned bounding boxes of the RankingAF for most of the fish video frames were even better than the ground-truth. On the contrary, both MDNet and ECO drifted from the target in the fish video and returned less accurate bounding boxes for the target object in the diving video

4.2. Hybrid Method Experiments

Although there are many datasets for training and testing single object tracking methods in videos such as LaSOT, OTB, TrackingNet [94], VOT and GOT-10k, there is no dataset consisting only of UAVs. For our specific UAV tracking application, a new dataset with various kinds of UAVs was created. To this end, we tracked UAVs using a camera mounted on another flying UAV. This dataset which called UAV Tracking, contained 15 video streams with 7500 frames which were high precision labeled. We used 2000 frames to train the YOLOv3 and YOLOv3-Tiny models, while the remaining 5500 frames were used for testing our approaches. In addition to these experiments, we also developed the generalized version of our approaches using MS-COCO pre-trained YOLOv3 models to compare with other trackers. Next, we conducted experiments on the UAV123 and UAV20L datasets consists of images captured by UAVs. We used the OTB

evaluation toolkit using OPE protocol for a fair comparison on all datasets. We report the results of methods using both success and precision plots. We named our tracker employing YOLOv3 as UAVH and employing YOLOv3-Tiny model as UAVH-Tiny. We conducted a small experiment on a validation dataset to determine the thresholds for detector and tracker and we obtained the thresholds of 0.5 and 0.45, respectively.

The proposed hybrid tracker was implemented in Python and the YOLOv3 models were trained on NVDIA Tesla V100 16GB GPU. We tested all experiments both on a notebook with an Intel Xeon E-2186 CPU and Nvidia P4200 8GB GPU and on a Jetson TX2 single-board computer.

4.2.1. UAV Tracking dataset

We trained and evaluated the proposed hybrid methods, UAVH and UAVH-Tiny on this created dataset. We also evaluated 7 different methods including YOLOv3, YOLOv3-Tiny, CSRT [92], TLD [16], MIL [18], BACF and KCF on the UAV Tracking dataset. Table 4.12 shows the results of our methods as well as other methods in terms of AUC, precision and FPS. Our trackers achieve the best results for both AUC and precision scores. As expected, UAVH scores better than UAVH-Tiny, while UAVH-Tiny is faster than UAVH. Our trackers, UAVH and UAVH-Tiny, also operated at 53.5 and 69.2 FPS, respectively.

Table 4.12. *Comparison of the proposed methods with other trackers in terms of AUC, Precision and FPS on the UAV Tracking dataset (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	AUC	Precision	FPS
UAVH	0.561	0.773	53.5
UAVH-Tiny	0.524	0.737	69.2
YOLOv3	0.485	0.653	16.5
YOLOv3Tiny	0.461	0.63	47.1
CSRT	0.232	0.402	111
TLD	0.205	0.265	20.1
MIL	0.215	0.379	12.3
KCF	0.126	0.196	115
BACF	0.213	0.339	25.8

As seen in the table, faster KCF and CSRT trackers achieved very low scores on this dataset. Results of the tested methods on UAV Tracking dataset created are given in Figure 4.5. The proposed methods also significantly outperformed YOLOv3 and KCF which are components of proposed methods.

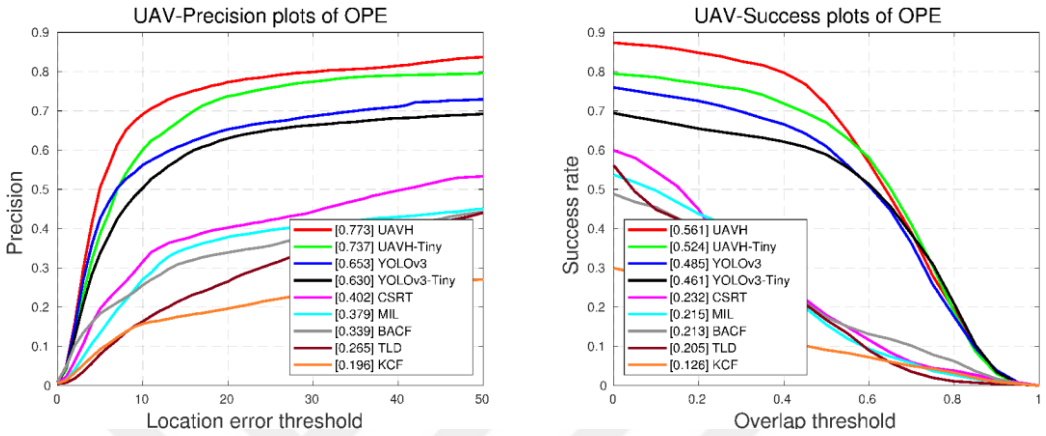


Figure 4.5. *Precision and success plots on the created UAV Tracking dataset*

4.2.2. Unmanned aerial vehicle 123 (UAV123) dataset

The UAV123 contains 123 low-altitude aerial video sequences (high-resolution) with 12 different challenging attributes captured by an UAV. Table 4.13 shows AUC and precision scores and operating times of the proposed and other trackers including ECO, ECO-HC, SRDCF [95], MEEM [96], IVT [7], MUSTER [97], DSST [98], Struct [99], ASLA [100], OAB [14], CSK [101], KCF, and TLD on the UAV123 benchmark. The precision and success scores of the tested methods are given in Table 4.13. Figure 4.6 shows precision and success plots of these trackers. The proposed methods, UAVH and UAVH-Tiny were pre-trained using MS-COCO dataset. The proposed UAVH achieved the best both precision of 74.3% and AUC of 52.7% scores.

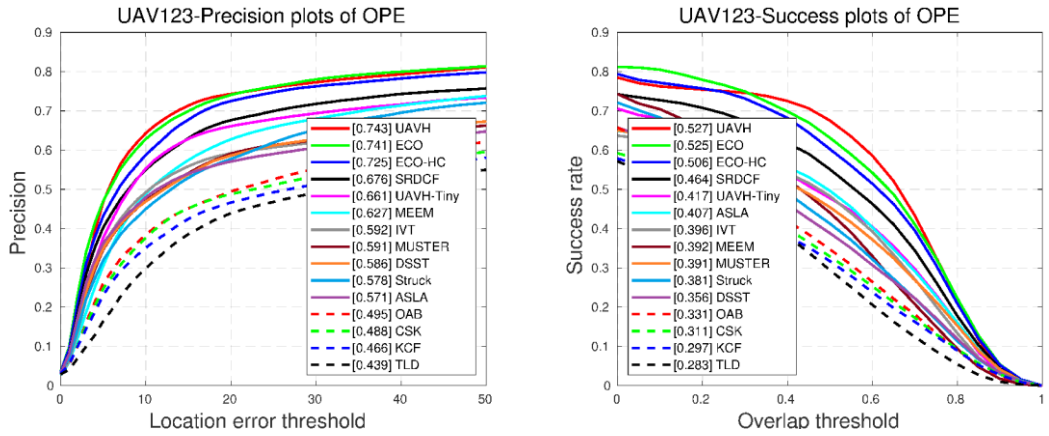


Figure 4.6. *Precision and success plots on the UAV123 dataset using the OPE protocol*

Table 4.13 also shows operating times of these trackers both on a PC and a Jetson TX2 onboard computer. As the table indicates, our hybrid trackers operate in real-time on both a computer and a Jetson TX2.

Table 4.13. *Comparison of the proposed methods with other trackers in terms of AUC and Precision on the UAV123 dataset (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	UAV123		FPS	
	AUC	**Precision**	**PC**	**Jetson TX2**
UAVH	0.527	0.743	62.86	25
UAVH-Tiny	0.417	0.661	71.9	34
ECO	0.525	0.741	8	-
ECO-HC	0.506	0.725	60	-
SRDCF	0.464	0.676	5.37	-
MUSTER	0.391	0.591	0.975	-
DSST	0.356	0.586	25.4	-
OAB	0.331	0.495	0.459	-
Struck	0.381	0.578	17	-
KCF	0.297	0.466	125	-
TLD	0.283	0.436	13.8	-

We also report the success plots in Figure 4.6 for 12 different attributes in the UAV123 dataset, including aspect ratio change, background clutter, camera motion, fast motion, full occlusion, illumination variation, low resolution, out-of-view, partial occlusion, similar object, scale variation and viewpoint change. The AUC scores of trackers are shown in the legend. The proposed UAVH obtained the best results in all

scenarios except background clutter, low resolution and similar object. Notably, in the aspect ratio change and fast motion scenarios, the proposed tracker, UAVH, significantly outperformed the second-best tracker, ECO, with relative gains of 12.7% and 17%, respectively thanks to the adaptability of the YOLO components. In the similar object, low resolution and background clutter scenarios, our methods failed to outperform ECO since target object was searched for by applying the YOLO detector in a ROI around the previous position. Our methods only fail to outperform other state-of-the-art trackers on camera motion since the whole scene moved with the camera in this challenge, and the motion network in the proposed methods did not discriminate the object movement from the background. Notably, for the similar object scenario, ECO tracker significantly outperformed UAVH with 5% gain in the AUC score since the detector YOLO was not designed to discriminate the objects in the same classes. Similarly, the low resolution and background clutter were difficult challenges for object detectors, and the detectors can fail under these scenarios.

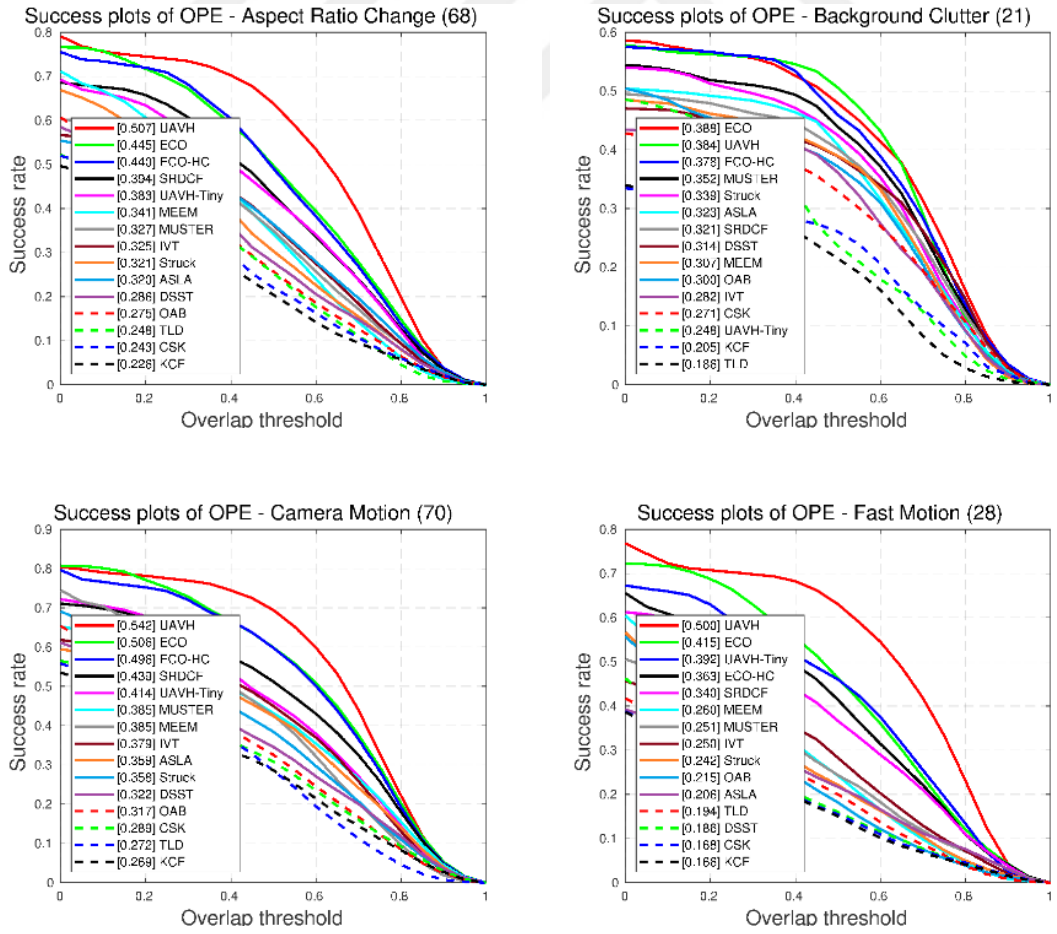


Figure 4.7. *Success plots for 12 different attributes on the UAV123 dataset*

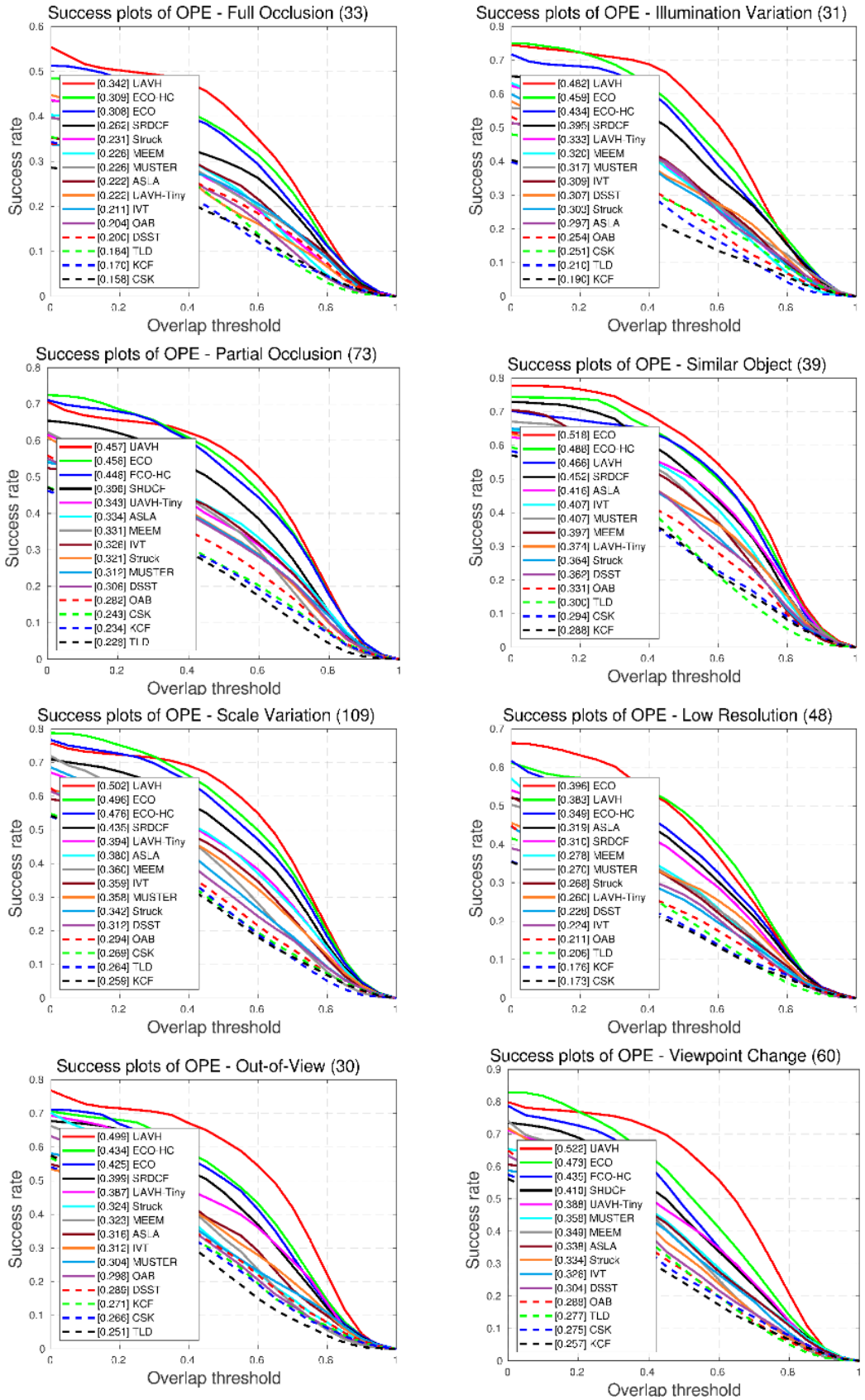


Figure 4.7. ***(continued)*** *Success plots for 12 different attributes on the UAV123 dataset*

4.2.3. UAV20L dataset

This dataset was a subset of the UAV123 benchmark designed to test for long-term aerial single object tracking. It includes 20 video streams with over 58,000 frames. The precision and success plots are shown in Figure 4.8. The proposed trackers, UAVH and UAVH-Tiny, achieved the best two results on this dataset. In terms of AUC, UAVH significantly outperformed UAVH-Tiny, while UAVH-Tiny achieved the second-best precision score of 76.1%, competitive with UAVH. The AUC and precision scores with operating time are given in Table 4.14. In this long-term dataset, UAVH-Tiny obtained better performances than a short-term dataset due to the self-correction mechanism. Because when the tracker fails, which is more likely to occur with long-term videos, our detector runs to redetect the target object.

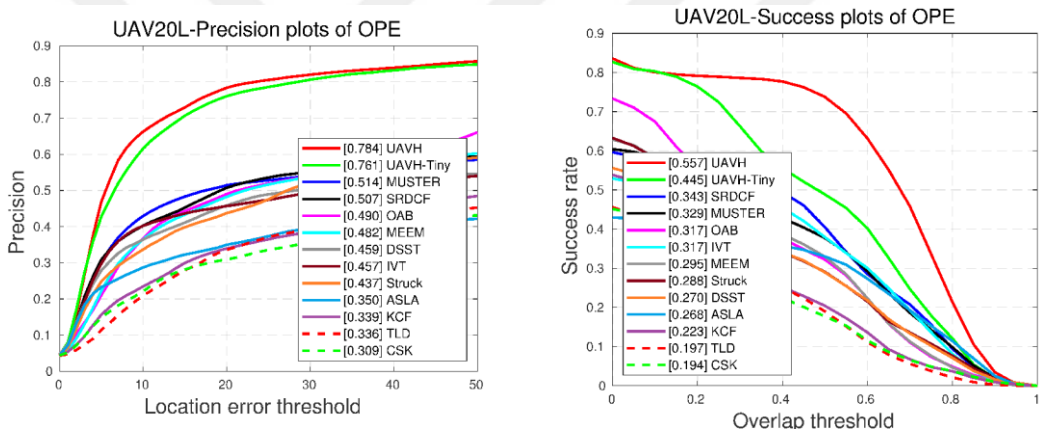


Figure 4.8. *Precision and success plots on the UAV20L dataset*

Table 4.14. *Comparison of the proposed methods with other trackers in terms of AUC, Precision and FPS on the UAV20L dataset (The red, blue and green fonts indicate the top-performing trackers, respectively)*

	UAV20L		FPS	
	AUC	**Precision**	**PC**	**Jetson TX2**
UAVH	0.557	0.784	62.86	25
UAVH-Tiny	0.445	0.761	71.9	34
ECO	0.435	0.604	8	-
SRDCF	0.343	0.507	5.37	-
MUSTER	0.329	0.514	0.975	-
DSST	0.27	0.459	25.4	-
OAB	0.317	0.490	0.459	-
Struck	0.288	0.437	17	-
KCF	0.223	0.339	125	-
TLD	0.197	0.336	13.8	-

4.2.4. Visual comparison

Finally, in Image 4.2, we present a visual comparison of proposed methods UAVH and UAVH-Tiny with other methods including CSRT and KCF, on the UAV Tracking dataset created. As seen in the image, the proposed methods both return more precise bounding boxes and show more robustness tracking performances.

Image 4.2. *A comparison of the proposed methods, UAVH (red bounding boxes), UAVH-Tiny (blue bounding boxes) with other trackers including CSRT (green bounding boxes), KCF (yellow bounding boxes) on the UAV Tracking dataset (Black bounding boxes shows ground-truth)*

5. CONCLUSION

This thesis introduces two different single object tracking methods which are deep learning-based and correlation filter-based. First, we developed a general single object tracker, RankingT, to obtain state-of-the-art results on different datasets. Next, we developed a hybrid tracker, UAVH, to detect and track UAVs.

In the RankingT tracker, we employed architecture of the MDNet tracker. We added a number of components, namely ranking loss, hard mining sampling strategy, and two-stream network to improve tracking performances. Then, we investigated the impacts of these components on four different datasets. Finally, we compared our final two-stream trackers with state-of-the-art trackers on eight different datasets. More precisely, we used hinge loss instead of MDNet loss function which employed soft-max loss. In addition to this hinge loss function, we proposed a novel ranking loss function. To this end, ranking pairs with different *IoU* ratios, which are greater than 0.1, between pairs and predicted target positions were selected randomly. We ensured that the classifier gave higher scores to the candidate regions that frame the target object better. We also used hard ranking mining strategy to improve performance of the proposed ranking loss term. In this strategy, selected ranking pairs were not matched randomly, and to use ranking pairs more effectively, the ranking loss was calculated between low-scoring samples with high *IoU* ratios and high-scoring samples with low *IoU* ratios. Lastly, we introduced a lightweight two-stream deep neural network method to track our object using spatial and temporal information in videos. To extract temporal information, we used motion information by calculating optical flows between consecutive frames, while using RGB images to extract spatial information. We designed two different lightweight networks which consist of 3 *conv* layers and 3 *fc* layers. We proposed two different strategies to combine two-stream networks: feature fusion (early fusion) and classifier output (score) fusion (late fusion). In the early fusion strategy, overall accuracy reduced. We decided to fuse the scores of these networks (late fusion) to improve the performances of our tracker. To this end, we proposed two different approaches: average score fusion and class-specific score fusion.

The proposed methods, RankingAF and RankingSF, obtained the best or competitive results on eight different datasets including OTB-2015, TColor-128, UAV123, DTB70, GOT-10k, LaSOT, NfS and VOT2017/2018. The performances of the

proposed methods are especially very promising on the OTB-2015, NfS, DTB70 and TColor-128 datasets.

We also conducted ablation analysis on four different datasets including OTB-2015, TColor-128, DTB70, and VOT-2017/2018 to investigate the importance of the proposed components. The results show that the proposed ranking loss term with hard sample mining strategy significantly outperforms MDNet and RankingT, which is without ranking loss term trackers. Additionally, proposed two-stream approaches improve tracker performances on the TColor-128, DTB70 and VOT2017/2018 datasets.

To track UAVs in images captured by another UAV in real-time, we developed another correlation filter-based tracking algorithm, UAVH. In the UAVH tracker which is a correlation filter-based real-time tracker, we proposed a hybrid detection-based tracker using the YOLOv3 detector and KCF tracker to reduce limitations and to take advantage of these methods. A new UAV dataset that consisted only of certain kinds of UAVs was created to train and test trackers. This dataset includes 15 videos with 7500 frames. We trained the YOLOv3 model using these UAV images to detect UAVs. In addition to the YOLOv3 model, we trained the YOLOv3-Tiny model to obtain a faster tracker. Next, we generalized our tracker for general object tracking to compare with other trackers. To this end, we used COCO dataset to train YOLOv3 and YOLOv3-Tiny models. We tested our final UAVH tracker on UAV123 and UAV20L datasets.

The proposed trackers, UAVH and UAVH-Tiny, achieved the best two results on the created UAV Tracking dataset and also provided real-time performances of 53.5 and 69.2 FPS, respectively, on a single board computer. Generalized version of proposed tracker UAVH achieved the best AUC and precision scores on UAV123 dataset. UAVH and UAVH-Tiny obtain the best two results on UAV20L dataset for both AUC and precision scores, respectively. These trackers ran in real-time on the UAV123 and UAV20L datasets. As expected, UAVH scored better than UAVH-Tiny, while UAVH-Tiny was faster than UAVH.

REFERENCES

[1] Li, F., Yao, Y., Li, P., Zhang, D., Zuo, W., and Yang, M.H. (2017). Integrating boundary and center correlation filters for visual tracking with aspect ratio variation. In: *Proceeding IEEE International Conference on Computer Vision Workshops*, pp. 2001-2009.

[2] Tan, Z., Nie, X., Qian, Q., Li, N., and Li, H. (2019). Learning to rank proposals for object detection. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 8273-8281.

[3] Nam, H., and Han, B. (2016). Learning multi-domain convolutional neural networks for visual tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4293-4302.

[4] Redmon, J., and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[5] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine İntelligence*, *37* (3), 583-596.

[6] Zhang, T., Ghanem, B., Liu, S., and Ahuja, N. (2012). Low-rank sparse learning for robust visual tracking. In: *Proceeding European Conference on Computer Vision*, pp. 470-484.

[7] Ross, D.A., Lim, J., Lin, R.S., and Yang, M.H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, *77* (1-3), 125-141.

[8] Wang, D., Lu, H., and Yang, M.H. (2012). Online object tracking with sparse prototypes. *IEEE Transactions on Image Processing*, *22* (1), 314-325.

[9] Han, B., Comaniciu, D., Zhu, Y., and Davis, L.S. (2008). Sequential kernel density approximation and its application to real-time visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30* (7), 1186-1197.

[10] Jepson, A.D., Fleet, D.J., and El-Maraghi, T.F. (2003). Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25* (10), 1296-1311.

[11] Held, D., Thrun, S., and Savarese, S. (2016). Learning to track at 100 fps with deep regression networks. In: *Proceeding European Conference on Computer Vision*, pp. 749-765.

[12] Mei, X., and Ling, H. (2009). Robust visual tracking using ℓ1 minimization. In: *Proceeding IEEE 12th International Conference on Computer Vision*, pp. 1436-1443.

[13] Zhang, T., Ghanem, B., Liu, S., and Ahuja, N. (2012). Robust visual tracking via multi-task sparse learning. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2042-2049.

[14] Grabner, H., Grabner, M., and Bischof, H. (2006). Real-time tracking via on-line boosting. In: *Proceeding* British Machine Vision Conference (*BMVC)*, 1 (5), pp. 6.

[15] Avidan, S. (2007). Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29* (2), 261-271.

[16] Kalal, Z., Mikolajczyk, K., and Matas, J. (2011). Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34* (7), 1409-1422.

[17] Avidan, S. (2004). Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26* (8), 1064-1072.

[18] Babenko, B., Yang, M.H., and Belongie, S. (2010). Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33* (8), 1619-1632.

[19] Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys (CSUR)*, *38* (4), 13.

[20] Wu, Y., Lim, J., and Yang, M.H. (2013). Online object tracking: A benchmark. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2411-2418.

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

[22] Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., and Torr, P.H. (2016). Fully-convolutional siamese networks for object tracking. In: *Proceeding European Conference on Computer Vision*, pp. 850-865.

[23] Bolme, D.S., Beveridge, J.R., Draper, B.A., and Lui, Y.M. (2010). Visual object tracking using adaptive correlation filters. In: *Proceeding IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2544-2550.

[24] Danelljan, M., Bhat, G., Shahbaz Khan, F., and Felsberg, M. (2017). ECO: Efficient convolution operators for tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6638-6646.

[25] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceeding IEEE conference on computer vision and pattern recognition*, pp. 580-587.

[26] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*.

[27] Jung, I., Son, J., Baek, M., and Han, B. (2018). Real-time MDNet. In: *Proceeding European Conference on Computer Vision,* pp. 83-98.

[28] Girshick, R. (2015). Fast R-CNN. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 1440-1448.

[29] Wang, N., and Yeung, D.Y. (2013). Learning a deep compact image representation for visual tracking. In: *Proceeding Advances in Neural Information Processing Systems*, pp. 809-817.

[30] Wang, N., Li, S., Gupta, A., and Yeung, D.Y. (2015). Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*.

[31] Hong, S., You, T., Kwak, S., and Han, B. (2015). Online tracking by learning discriminative saliency map with convolutional neural network. In: *Proceeding International Conference on Machine Learning,* pp. 597-606.

[32] Li, H., Li, Y., and Porikli, F. (2015). Deeptrack: Learning discriminative feature representations online for robust visual tracking. *IEEE Transactions on Image Processing*, *25* (4), 1834-1848.

[33] Danelljan, M., Shahbaz Khan, F., Felsberg, M., and Van de Weijer, J. (2014). Adaptive color attributes for real-time visual tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1090-1097.

[34] Ma, C., Huang, J.B., Yang, X., and Yang, M.H. (2015). Hierarchical convolutional features for visual tracking. In*: Proceeding IEEE International Conference on Computer Vision,* pp. 3074-3082.

[35] Danelljan, M., Hager, G., Shahbaz Khan, F., and Felsberg, M. (2015). Convolutional features for correlation filter based visual tracking. In: *Proceeding IEEE International Conference on Computer Vision Workshops,* pp. 58-66.

[36] Valmadre, J., Bertinetto, L., Henriques, J., Vedaldi, A., and Torr, P.H. (2017). End-to-end representation learning for correlation filter based tracking. In*: Proceeding IEEE Conference on Computer Vision and Pattern Recognition,* pp. 2805-2813.

[37] Wang, Q., Gao, J., Xing, J., Zhang, M., and Hu, W. (2017). DCFNet: Discriminant correlation filters network for visual tracking. *arXiv preprint arXiv:1704.04057*.

[38] Gundogdu, E., and Alatan, A.A. (2018). Good features to correlate for visual tracking. *IEEE Transactions on Image Processing*, *27* (5), 2526-2540.

[39] Danelljan, M., Robinson, A., Khan, F.S., and Felsberg, M. (2016). Beyond correlation filters: Learning continuous convolution operators for visual tracking. In*: Proceeding European Conference on Computer Vision*, pp. 472-488.

[40] Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., Cehovin Zajc, L., and Fernandez, G. (2017). The visual object tracking VOT2017 challenge results. In*: Proceeding IEEE International Conference on Computer Vision Workshops*, pp. 1949-1972.

[41] Wang, N., Song, Y., Ma, C., Zhou, W., Liu, W., and Li, H. (2019). Unsupervised deep tracking. In*: Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1308-1317.

[42] Xu, T., Feng, Z.H., Wu, X.J., and Kittler, J. (2019). Joint group feature selection and discriminative filter learning for robust visual object tracking. In*: Proceeding IEEE International Conference on Computer Vision,* pp. 7950-7960.

[43] Dai, K., Wang, D., Lu, H., Sun, C., and Li, J. (2019). Visual tracking via adaptive spatially-regularized correlation filters. In*: Proceeding IEEE Conference on Computer Vision and Pattern Recognition,* pp. 4670-4679.

[44] Huang, Z., Fu, C., Li, Y., Lin, F., and Lu, P. (2019). Learning aberrance repressed correlation filters for real-time UAV tracking. In*: Proceeding IEEE International Conference on Computer Vision,* pp. 2891-2900.

[45] Mueller, M., Smith, N., and Ghanem, B. (2017). Context-aware correlation filter tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1396-1404.

[46] Liu, S., Zhang, T., Cao, X., and Xu, C. (2016). Structural correlation filter for robust visual tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4312-4320.

[47] Tao, R., Gavves, E., and Smeulders, A.W. (2016). Siamese instance search for tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1420-1429.

[48] Wang, B., Wang, L., Shuai, B., Zuo, Z., Liu, T., Luk Chan, K., and Wang, G. (2016). Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition Workshops,* pp. 1-8.

[49] Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., and Wang, S. (2017). Learning dynamic siamese network for visual object tracking. In: *Proceeding IEEE International Conference on Computer Vision,* pp. 1763-1771.

[50] Li, B., Yan, J., Wu, W., Zhu, Z., and Hu, X. (2018). High performance visual tracking with siamese region proposal network. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition,* pp. 8971-8980.

[51] Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., and Yan, J. (2019). SiamRPN++: Evolution of siamese visual tracking with very deep networks. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition,* pp. 4282-4291.

[52] Fan, H., and Ling, H. (2019). Siamese cascaded region proposal networks for real-time visual tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition,* pp. 7952-7961.

[53] Li, P., Chen, B., Ouyang, W., Wang, D., Yang, X., and Lu, H. (2019). Gradnet: Gradient-guided network for visual object tracking. In: *Proceeding IEEE International Conference on Computer Vision,* pp. 6162-6171.

[54] Wang, Q., Zhang, L., Bertinetto, L., Hu, W., and Torr, P.H. (2019). Fast online object tracking and segmentation: A unifying approach. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1328-1338.

[55] Choi, J., Kwon, J., and Lee, K.M. (2019). Deep meta learning for real-time target-aware visual tracking. In: *Proceeding IEEE International Conference on Computer Vision,* pp. 911-920.

[56] Wang, G., Luo, C., Xiong, Z., and Zeng, W. (2019). Spm-tracker: Series-parallel matching for real-time visual object tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3643-3652.

[57] Danelljan, M., Bhat, G., Khan, F.S., and Felsberg, M. (2019). ATOM: Accurate tracking by overlap maximization. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4660-4669.

[58] Kristan, M., Matas, J., Leonardis, A., Felsberg, M., Pflugfelder, R., Kamarainen, J.K., and Eldesokey, A. (2019). The seventh visual object tracking VOT2019 challenge results. In: *Proceeding IEEE International Conference on Computer Vision Workshops*, pp. 0-0.

[59] Bhat, G., Danelljan, M., Gool, L.V., and Timofte, R. (2019). Learning discriminative model prediction for tracking. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 6182-6191.

[60] Huang, L., Zhao, X., and Huang, K. (2019). Bridging the gap between detection and tracking: A unified approach. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 3999-4009.

[61] Simonyan, K., and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In: *Proceeding Advances in Neural Information Processing Systems*, pp. 568-576.

[62] Weinzaepfel, P., Harchaoui, Z., and Schmid, C. (2015). Learning to track for spatio-temporal action localization. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 3164-3172.

[63] Narayan, S., Cholakkal, H., Khan, F.S., and Shao, L. (2019). 3C-Net: Category count and center loss for weakly-supervised action localization. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 8679-8687.

[64] Köpüklü, O., Wei, X., and Rigoll, G. (2019). You Only Watch Once: A Unified CNN Architecture for Real-Time Spatiotemporal Action Localization. *arXiv preprint arXiv:1911.06644*.

[65] Jiang, B., Wang, M., Gan, W., Wu, W., and Yan, J. (2019). STM: Spatiotemporal and motion encoding for action recognition. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 2000-2009.

[66] Gladh, S., Danelljan, M., Khan, F.S., and Felsberg, M. (2016). Deep motion features for visual tracking. *In: Proceeding 23rd International Conference on Pattern Recognition (ICPR)*, pp. 1243-1248.

[67] Zhu, Z., Wu, W., Zou, W., and Yan, J. (2018). End-to-end flow correlation tracking with spatial-temporal attention. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 548-557.

[68] Teng, Z., Xing, J., Wang, Q., Lang, C., Feng, S., and Jin, Y. (2017). Robust object tracking based on temporal and spatial deep networks. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 1144-1153.

[69] Gao, J., Zhang, T., and Xu, C. (2019). Graph convolutional tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4649-4659.

[70] Zhang, K., Zhang, L., Liu, Q., Zhang, D., and Yang, M. H. (2014). Fast visual tracking via dense spatio-temporal context learning. In: *Proceeding European Conference on Computer Vision*, pp. 127-141.

[71] https://cs231n.github.io/convolutional-networks/, (accessed: 11.07.2020).

[72] Noroozi, M., and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In: *Proceeding European Conference on Computer Vision*, pp. 69-84.

[73] Lee, H.Y., Huang, J.B., Singh, M., and Yang, M.H. (2017). Unsupervised representation learning by sorting sequences. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 667-676.

[74] Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In: *Proceeding European Conference on Computer Vision*, pp. 25-36.

[75] Zitnick, C. L., and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In: *Proceeding European Conference on Computer Vision*, pp. 391-405.

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

[77] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., and Berg, A.C. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115* (3), 211-252.

[78] Cevikalp, H., and Triggs, B. (2017). Visual object detection using cascades of binary and one-class classifiers. *International Journal of Computer Vision*, *123* (3), 334-349.

[79] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Proceeding Advances in Neural Information Processing Systems*, pp. 91-99.

[80] Liang, P., Blasch, E., and Ling, H. (2015). Encoding color information for visual tracking: Algorithms and benchmark. *IEEE Transactions on Image Processing*, *24* (12), 5630-5644.

[81] Mueller, M., Smith, N., and Ghanem, B. (2016). A benchmark and simulator for uav tracking. In: *Proceeding European Conference on Computer Vision*, pp. 445-461.

[82] Li, S., and Yeung, D.Y. (2017). Visual object tracking for unmanned aerial vehicles: A benchmark and new motion models. In: *Proceeding* 31st *AAAI Conference on Artificial Intelligence,* pp. 4140-4146.

[83] Kiani Galoogahi, H., Fagg, A., Huang, C., Ramanan, D., and Lucey, S. (2017). Need for speed: A benchmark for higher frame rate object tracking. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 1125-1134.

[84] Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., and Ling, H. (2019). LaSOT: A high-quality benchmark for large-scale single object tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5374-5383.

[85] Huang, L., Zhao, X., and Huang, K. (2019). Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 1-17.

[86] Li, X., Ma, C., Wu, B., He, Z., and Yang, M.H. (2019). Target-aware deep tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1369-1378.

[87] Song, Y., Ma, C., Wu, X., Gong, L., Bao, L., Zuo, W., and Yang, M.H. (2018). Vital: Visual tracking via adversarial learning. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8990-8999.

[88] Sun, C., Wang, D., Lu, H., and Yang, M.H. (2018). Learning spatial-aware regressions for visual tracking. In*: Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8962-8970.

[89] Zhang, T., Xu, C., and Yang, M.H. (2017). Multi-task correlation particle filter for robust object tracking. In*: Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4335-4343.

[90] Kiani Galoogahi, H., Fagg, A., and Lucey, S. (2017). Learning background-aware correlation filters for visual tracking. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 1135-1143.

[91] He, Z., Fan, Y., Zhuang, J., Dong, Y., and Bai, H. (2017). Correlation filters with weighted convolution responses. In: *Proceeding IEEE International Conference on Computer Vision Workshops*, pp. 1992-2000.

[92] Lukezic, A., Vojir, T., Cehovin Zajc, L., Matas, J., and Kristan, M. (2017). Discriminative correlation filter with channel and spatial reliability. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6309-6318.

[93] Kokul, T., Fookes, C., Sridharan, S., Ramanan, A., and Pinidiyaarachchi, U.A.J. (2017). Gate connected convolutional neural network for object tracking. In: *Proceeding IEEE International Conference on Image Processing (ICIP)*, pp. 2602-2606.

[94] Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., and Ghanem, B. (2018). Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In: *Proceeding European Conference on Computer Vision*, pp. 300-317.

[95] Danelljan, M., Hager, G., Shahbaz Khan, F., and Felsberg, M. (2015). Learning spatially regularized correlation filters for visual tracking. In: *Proceeding IEEE International Conference on Computer Vision*, pp. 4310-4318.

[96] Zhang, J., Ma, S., and Sclaroff, S. (2014). MEEM: robust tracking via multiple experts using entropy minimization. In: *Proceeding European Conference on Computer Vision*, pp. 188-203.

[97] Hong, Z., Chen, Z., Wang, C., Mei, X., Prokhorov, D., and Tao, D. (2015). Multi-store tracker (MUSTER): A cognitive psychology inspired approach to object tracking. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 749-758.

[98] Danelljan, M., Häger, G., Khan, F.S., and Felsberg, M. (2016). Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39* (8), 1561-1575.

[99] Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M.M., Hicks, S.L., and Torr, P.H. (2015). Struck: Structured output tracking with kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38* (10), 2096-2109.

[100] Jia, X., Lu, H., and Yang, M.H. (2012). Visual tracking via adaptive structural local sparse appearance model. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1822-1829.

[101] Henriques, J.F., Caseiro, R., Martins, P., and Batista, J. (2012). Exploiting the circulant structure of tracking-by-detection with kernels. In: *Proceeding IEEE European Conference on Computer Vision*, pp. 702-715.

CURRICULUM VITAE

ORCID ID: 0000-0001-8709-9872

Name:	Hasan Saribaş
Foreign Language:	English
Birth date and place:	15/08/1988
E-mail:	hasansaribas@eskisehir.edu.tr

Education

- PhD (2015-2020), Eskisehir Technical University, Institute of Graduate Programs, Department of Avionics
- MSc (2013-2015), Anadolu University, Graduate School of Sciences, Department of Avionics
- BSc (2007-2011), Atatürk University, Department of Electrical-Electronics Engineering

Employment

- Research Assistant (2018-), Eskisehir Technical University, Department of Avionics
- Research Assistant (2013-2018), Anadolu University, Department of Avionics

Publications

- Cevikalp, H., Saribas, H., Benligiray, B., and Kahvecioglu, S. (2019). Visual Object Tracking by Using Ranking Loss. In: *Proceeding IEEE International Conference on Computer Vision Workshops* (pp. 0-0).
- Uzun, B., Eker, O., Saribaş, H., and Çevikalp, H. (2019, April). Detection Based Tracking of Unmanned Aerial Vehicles. In: *Proceeding 27th Signal*

Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

- Cevikalp, H., Benligiray, B., Gerek, Ö. N., and Saribas, H. (2019). Semi-Supervised Robust Deep Neural Networks for Multi-Label Classification. In: *Proceeding CVPR Workshops* (pp. 9-17).
- Saribas, H., Uzun, B., Benligiray, B., Eker, O., and Cevikalp, H. (2019). A Hybrid Method for Tracking of Objects by UAVs. In: *Proceeding IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 0-0).
- Sarıbaş, H., Tatli, A., and Onrat, A. (2018, May). Control of unmanned aerial vehicles using self tuning fuzzy PID. In: *Proceeding 26th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE.
- Sarıbaş, H., Çevıkalp, H., and Kahvecıoğlu, S. (2018, May). Car detection in images taken from unmanned aerial vehicles. In: *Proceeding 26th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE.
- Saribaş, H. (2015). *Dört rotorlu bir insansız hava aracının kesir dereceli denetleyici ile kontrolü* (Master's thesis, Anadolu Üniversitesi).
- Saribas, H., and Kahvecioglu, S. (2015). Control of quadrotor using particle swarm optimization tuned fractional order PID controller. In: *Proceeding 8th Ankara Interantional Aerospace Conference* (pp. 10-12).
- Saribas, H., Cevikalp, H., and Kahvecioglu, S. Car Localization in Aerial Images Taken from Quadcopter. In: *Proceeding 14th International Conference on Recent Trends in Engineering and Technology (RTET).*
- Saribas, H.and Kahvecioglu, S. (2014) Kesirli Denetleyici ile Dört Rotorlu İnsansız Hava Aracının Kontrolü. In: *Proceeding Otomatik Kontrol Türk Milli Komitesi (TOK2014).*

- Ermeydan, A., Yıldız, E. and Saribas, H. (2013). Bir Uçağın İstenilen Baş Açısında Koordineli Dönüş Tasarımında Klasik ve Bulanık Mantığın Karşılaştırılması. *Ulusal Havacılık Teknolojisi ve Uygulamaları Kongresi (UHAT)*.
- Ermeydan, A., Yıldız, E. and Saribas, H. (2013). Bir uçağın İstenilen Uçuş Baş Açısına Koordineli Dönüş Tasarımı. *Otomatik Kontrol Türk Milli Komitesi (TOK2013)*.

Awards

- Alper Atalay-The second best student paper, (2019), *27th Signal Processing and Communications Applications Conference (SIU)*.
- Tubitak-Bideb National Scholarship Programme for PhD Students (2015).
- Tubitak-Bideb National Scholarship Programme for MSc Students (2012).