Visual Object Tracking in Dynamic Scenes

by

© Mohamed Hamed Abdelpakey

A thesis submitted to the School of Graduate Studies
in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

February 2021

St. John's, Newfoundland

Abstract

Visual object tracking is a fundamental task in the field computer vision. Visual object tracking is widely used in numerous applications which include, but are not limited to video surveillance, image understanding, robotics, and human-computer interaction. In essence, visual object tracking is the problem of estimating the states/trajectory of the object of interest over time. Unlike other tasks such as object detection where the number of classes/categories are defined beforehand, the only available information of the object of interest is at the first frame.

Even though, Deep Learning (DL) has revolutionised most computer vision tasks, visual object tracking still imposes several challenges. The nature of visual object tracking task is stochastic, where no prior-knowledge is available about the object of interest during the training or testing/inference. Moreover, visual object tracking is a class-agnostic task, as opposed object detection and segmentation tasks. In this thesis, the main objective is to develop and advance the visual object trackers using novel designs of deep learning frameworks and mathematical formulations.

To take advantage of different trackers, a novel framework is developed to track moving objects based on a composite framework and a reporter mechanism. The composite framework has built-in trackers and user-defined trackers to track the object of interest. The framework contains a module to calculate the robustness for each tracker and a reporter mechanism serves as a recovery mechanism if trackers fail to locate the object of interest.

Different trackers may fail to track the object of interest, thus, a more robust framework based on Siamese network architecture, namely DensSiam, is proposed to use the concept of dense layers and connects each dense layer in the network to all layers in a feed-forward fashion with a similarity-learning function. DensSiam also includes a Self-Attention mechanism to force the network to pay more attention to non-local features during offline training.

Generally, Siamese trackers do not fully utilize semantic and objectness information from pre-trained networks that have been trained on an image classification task. To solve this problem a novel architecture design is proposed , dubbed DomainSiam, to learn a Domain-Aware that fully utilizes semantic and objectness information while producing a class-agnostic track using a ridge regression network. Moreover, to reduce the sparsity problem, we solve the ridge regression problem with a differentiable weighted-dynamic loss function.

Siamese trackers have high speed and work in real-time, however, they lack high accuracy. To overcome this challenge, a novel dynamic policy gradient Agent-Environment architecture with Siamese network (DP-Siam) is proposed to train the tracker to increase the accuracy and the expected average overlap while running in real-time. DP-Siam is trained offline with reinforcement learning to produce a continuous action that predicts the optimal object location.

One of the common design block in most object trackers in the literature is the backbone network, where the backbone network is trained in the feature space. To design a backbone network that maps from feature space to another space (i.e., joint-nullspace) and more suitable for object tracking and classification, a novel framework is proposed. The new framework is called NullSpaceNet has a clear interpretation for the feature representation and the features in this space are more separable. NullSpaceNet is utilized in object tracking by regularizing the discriminative joint-nullspace backbone network. The novel tracker is called NullSpaceRDAR, and encourages the network to have a representation for the target-specific information for the object of interest in the joint-nullspace. In contrast to feature space where objects from a specific class are categorized into one category however, it is insensitive to intra-class variations.

Furthermore, we use the NullSpaceNet backbone to learn a tracker, dubbed NullSpaceRDAR, with a regularized discriminative joint-nullspace backbone network that is specifically designed for object tracking. In the regularized discriminative joint-nullspace, the features from the same target-specific are collapsed into one point in the joint-null space and different target-specific features are collapsed into different points in the joint-nullspace. Consequently, the joint-nullspace forces the network to be sensitive to the variations of the object from the same class (intra-class variations). Moreover, a dynamic adaptive loss function is proposed to select the suitable loss function from a super-set family of losses based on the training data to make NullSpaceRDAR more robust to different challenges.

To the soul of my father ...

To my caring mother and siblings ...

Acknowledgements

After thanking Almighty ”ALLAH” for his blessing and guidance to complete this work, I would like to offer my sincere thanks to my supervisor Dr. Mohamed Shehata for his continued and valuable guidance. I greatly appreciate the time he has spent contributing to this research and to my professional development.

I would like to acknowledge the financial support provided by my supervisor, the Faculty of Engineering and Applied Science, the School of Graduate Studies, and the Natural Sciences and Engineering Research Council of Canada (NSERC).

I would like to thank the staff of the Department of Electrical and Computer Engineering. I thank all my group members and lab colleagues; it was fun working with all of you.

I would like to acknowledge the most impactful persons in my life, my parents. I would have never been in this position without your unconditional support, love, and care.

Co-Authorship Statements

I, Mohamed Hamed Abdepakey, hold a principal author status for all the manuscript chapters (Chapter 2 - 7) in this dissertation. However, each manuscript is co-authored by my supervisor and co-researchers, whose contributions have facilitated the development of this work as described below.

- Paper 1 in Chapter 2:

  M. H. Abdelpakey, M. S. Shehata, M. M. Mohamed, and M. Gong, "Adaptive framework for robust visual tracking," IEEE Access, vol. 6,pp. 55273–55283, 2018.

  I was the primary author, with authors 2-4 contributing to the idea, its formulation and development, and refinement of the presentation.

- Paper 2 in Chapter 3:

  M. H. Abdelpakey, M. S. Shehata, and M. M. Mohamed, "DensSiam: End-to-end densely siamese network with self-attention model for object tracking," in International Symposium on Visual Computing. Springer, 2018, pp. 463–473. (Oral presentation, the method and results in this paper ranked 14^{th} out of 72 papers participated from all over the world in international competition organized and published by VOT community in ECCV2018)

  I was the primary author, with authors 2 and 3 contributing to the idea, its formulation and development, and refinement of the presentation.

- Paper 3 in Chapter 4:

  M. H. Abdelpakey and M. S. Shehata, "Domainsiam: Domain-aware siamese network for visual object tracking," in International Symposium on Visual Computing. Springer, 2019, pp. 45–58. (Oral presentation, nominated for the best paper)
  I was the primary author, with author 2 contributing to the idea, its formulation and development, and refinement of the presentation.

- Paper 4 in Chapter 5: M. H. Abdelpakey and M. S. Shehata,"Dp-siam: Dynamic policy siamese network for robust object tracking, " IEEE Transactions on Image Processing, vol. 29, pp. 1479–1492, 2019.
  I was the primary author, with author 2 contributing to the idea, its formulation and development, and refinement of the presentation.

- Paper 5 in Chapter 6: M. H. Abdelpakey and M. S. Shehata,"NullSpaceNet: Nullspace Convoluional Neural Network with Differentiable Loss Function" IEEE Transactions on Pattern Analysis and Machine Intelligenc (TPAMI), submitted.
  I was the primary author, with author 2 contributing to the idea, its formulation and development, and refinement of the presentation.

- Paper 6 in Chapter 7: M. H. Abdelpakey and M. S. Shehata,"NullSpaceRDAR: Regualrized Discriminative Adaptive NullSpace for Robust Visual Object Tracking", IEEE Transactions on Image Processing, submitted.
  I was the primary author, with author 2 contributing to the idea, its formulation and development, and refinement of the presentation.

Mohamed Abdelpakey

Mohamed Hamed Abdelpakey

Jan 3, 2021

Date

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background, Research Motivation and Contribution

The work in this thesis is mainly based on Siamese network. Siamese formulates the object tracking problem as a similarity metric learning for prediction. The network consists of two branches, namely the target branch and the search branch. The target branch takes in the target image to produce a filter-like and the search branch takes in the test image and produces a feature map. The filter-like and the feature map are combined using a cross-correlation layer which produces a score map to locate the object. Visual object tracking has a different nature compared to fundamental tasks such as object detection and segmentation, where in object tracking the classes/categories are not defined beforehand. Moreover, the tracking of the object of interest is often done in an unconstrained environment. Motivated by these observations, different network designs using Deep Learning (DL) along with mathematical formulations that are specifically designed for visual object tracking are studied in this thesis. We have identified and addressed these research problems:

Research problems:

- **P1**- Design a composite framework that has the ability to fuse multiple trackers' prediction and can recover from tracking failures.
- **P2**- Design a novel network architecture to utilize the non-local features of the object of interest.
- **P3**- Design a novel network to fully utilize the semantic and objectness of the object of interest.
- **P4**- Design a novel network to balance the trade-off between the high-speed and accuracy to track the object of interest.
- **P5**-Develop a mathematical formulation in the deep learning framework to map from feature space to joint-nullspace.
- **P6**- Design a novel network to incorporate the joint-nullspace in the object tracking.

Objectives and Contributions:

- **Objective1:** Problem **P1** has been solved in chapter 2 by proposing a framework which contains two built-in trackers and optional user trackers plugins that can be used by the user to include additional trackers in the framework to handle specific scenarios. This allows more flexibility and generalizes the framework to be efficient in different applications. Moreover, a new mechanism, called a *reporter*, that intervene when there is a tracking drift and correct the object trajectory. Furthermore, a new metric was developed, which is named a virtual vector to be combined with trajectory analysis to calculate a more accurate robustness score.

- **Objective2:** DensSiam has been proposed in Chapter 3 to solve the Problem **P2**. A novel end-to-end deep Densely-Siamese architecture is proposed for object tracking. The new architecture can capture the non-local features which are robust to appearance changes. Additionally, it reduces the number of shared parameters between layers while building up deeper network compared to other existing Siamese-based architectures commonly used in current state-of-the-art trackers. Moreover, an effective response map based on Self-Attention module that boosts the DensSiam tracker performance . The response map has no-local features and captures the semantic information about the target object. The proposed network architecture tackles the vanishing-gradient problem and leverages feature reuse to improve the generalization capability.

- **Objective3:** A novel network architecture has been proposed in Chapter 4 to tackle the problem in **P3**. The proposed network architecture captures the Domain-Aware features with semantic and objectness information. The proposed network architecture enables the features to be robust to appearance changes. Moreover, it decreases the sparsity problem, as it produces the most important feature space. Consequently, it decreases the overhead calculations. in addition to the proposed network architecture, a differentiable weighted-dynamic domain loss function is developed specifically for visual object tracking to train the regression network to extract the domain channels that are activated by target category. The developed loss is monotonic with respect to its hyper-parameters, and this will be useful in case of high-dimensional data and non-convexity. Consequently, this will increase the performance of the tracker. The proposed architecture tackles the generalization capability from one domain to another domain (e.g., from ImageNet to VOT datasets).

- **Objective4:** To solve problem **P4**, DP-Siam has been proposed in Chapter 5. Dp-Siam is a novel dynamic Siamese Agent-Environment architecture that formulates the tracking problem with reinforcement learning. DP-Siam produces a continuous action that predicts the optimal object location. DP-Siam has a novel architecture that consists of three networks: an Agent network to predict the optimal state of the object being tracked, an Environment network to get the Q-value during the offline training phase to minimize the error of the loss function, and a Siamese network to produce a heat-map. The Environment network acts as a verifier to the action of the Agent network during online tracking. Secondly, the proposed network architecture allows the tracker to dynamically select the hyper-parameters in each frame instead of the traditional method of fixing their values for the entire dataset, which has not been done before. Finally, the design of the proposed architecture increases the generalization to other domains (e.g. from ImageNet to VOT datasets).

- **Objective5:** Problem **P5** has been tackled in Chapter 6. A novel Network (NullSpaceNet) that learns to map from the pixel-level image to a joint-nullspace. The formulation of NullSpaceNet ensures that the nullspace features from the same class are collapsed into a single point while the ones from different classes are collapsed into different points with high separation margins. NullSpaceNet is architecture-agnostic, which means that it can easily integrate different feature extractors in its architecture. To train the proposed network, a differentiable loss function is developed to effectively train NullSpaceNet. The proposed loss function is different from the standard categorical cross-entropy functions where the proposed loss function ensures that the within-class scatter matrix vanishes while maintaining a positive between-class scatter matrix. The differentiable loss function has a closed-form solution with no free-parameters. On top of that, the proposed NullSpaceNet has a clear interpretation of the learned features, both mathematically and geometrically.

- **Objective6:** Finally, **P6** has been solved in Chapter 7. Firstly, a novel formulation for the feature learning in the backbone network by projecting the feature onto a joint-nullspace. The joint-nullspace ensures that the same target-specific information is collapsed into one point in the learned space, while different target-specific information is collapsed into different points in the learned space. Secondly, the new formulation produces a high discriminative power due to the high separation margin among the different points in the learned space, while being extremely-low separation margin among the same target-specific information points in the learned space. Finally, a dynamic loss is proposed to adaptively switch between loss function based on the training data.

1.2 Organization of the Thesis

Chapter 2 motivates the research in visual object tracking and identifies the problems that are being tackled in this thesis followed by the contributions. Chapter 3 introduces DensSiam tracker, a novel network architecture that is specifically designed for object tracking to utilize the non-local features. DomainSiam tracker is introduced in chapter 4 to solve the problem of semantic and objectness information of the object of interest. In Chapter 5, the network design of DP-Siam tracker is explained. Chapter 6 introduces the NullSpaceNet network to map from feature space to joint-nullspace. This Chapter is an introduction to Chapter 7. Chapter 7 provides explanation to NullSpaceRDAR tracker and how to regularize the tracker to work in a highly discriminative power features. Finally Chapter 8 concludes the thesis and gives a future directions in visual object tracking.

References

[1] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[2] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, “Beyond correlation filters: Learning continuous convolution operators for visual tracking,” in *European Conference on Computer Vision*. Springer, 2016, pp. 472–488.

Chapter 2

Adaptive Framework for Robust Visual Tracking

2.1 abstract

Visual tracking is a difficult and challenging problem, for numerous reasons such as small object size, pose angle variations, occlusion, and camera motion. Object tracking has many real-world applications such as surveillance systems, moving organs in medical imaging and robotics. Traditional tracking methods lack a recovery mechanism that can be used in situations when the tracked objects drift away from ground truth. In this chapter, we propose a novel framework for tracking moving objects based on a composite framework and a reporter mechanism. The composite framework tracks moving objects using different trackers and produces pairs of forward/backward tracklets. A robustness score is then calculated for each tracker using its forward/backward tracklet pair to find the most reliable moving object trajectory. The reporter serves as the recovery mechanism to correct the moving object trajectory when the robustness

score is very low, mainly using a combination of particle filter and template matching. The proposed framework can handle partial and heavy occlusions; moreover, the structure of the framework enables integration of other user-specific trackers. Extensive experiments on recent benchmarks show that the proposed framework outperforms other current state-of-the-art trackers due to its powerful trajectory analysis and recovery mechanism, the framework improved the area under curve from 68.0% to 70.8% on OTB-100 benchmark.

2.2 Introduction

Object tracking is very important in many applications such as image understanding, robotics, surveillance, and human-computer interaction [1]. In the last decade, the development of satellite and unmanned aerial vehicle (UAV) has significantly increased. Remote sensing videos, especially aerial ones, have been widely used in surveillance because of their ability to provide full coverage of ground areas of interest. However, objects tracking in these videos is very challenging due to many factors including small objects size, illumination changes, pose angle variations, occlusion, background clutter, and camera motion.

Conventional trackers (e.g. [2–5]) produce objects trajectories between successive frames using a specific model that updates the objects locations in new frames. Tracking drift is a common problem in tracking objects using conventional trackers, where the new locations estimated by the tracker being used start to drift away from the true object locations. Tracking drift is more persistent in aerial videos in situations when there is consistent partial or full occlusions.

Generally, object trackers can be categorized into three categories [1, 6]: generative, discriminative, and composite trackers. Generative trackers track objects by searching for the image region that best matches a template or an appearance model [1]. For example, the histogram-based tracking method [7] relies on treating the weighted histogram of the current object image patch as a template, and the mean shift is used as an efficient search strategy. Black $et\ al.$ [8] used a subspace as an offline template and tracked the object under the optical flow framework. He $et\ al.$ [9] created a local sensitive histogram as a template, that is invariant to illumination, and an exhaustive search of the image patch with the similar local histogram is performed in the vicinity of the object. In [10], the distribution field is used to define the probability pixels over a grayscale image to construct a template for the object. Zoidi $et\ al.$ [11] employed the similarity over a color histogram and texture descriptors to locate the target. Generally, the generative trackers discussed above fail in situations when there are occlusions and usually cannot be recovered from tracking drifts.

Discriminative trackers deal with the object tracking problem as a binary classification problem to separate the foreground from the background. Discriminative methods exploit the visual information from the target of interest and the background. Avidan $et\ al$[12] used the support vector machine classifier (SVM) with optical flow. CCOT [13] is based on discrimination correlation filter (DCF) and utilized multi-resolution deep feature map. Henriques $et\ al.$ [14] used an analytic model of correlation filter for datasets of thousands of translated patches to utilize the circulant matrix and diagonalized it with the Discrete Fourier Transform. In [15], a neural network (MDNet) was used with shared layers and multiple branches of domain-specific layers, where each branch is responsible for binary classification and each domain corresponds to the training sequence.

Discriminative trackers have good performance over time, however, similar to generative trackers, they still suffer from the same drift problems when there are frequent occlusions. CFNet traker [16] used Siamese network and integrated a correlation filter layer to the network. In [17] a semantic branch added to Siamese network to capture more robust deep features of the object of interest. In [18] a deconvolutional network is used as a learnable upsampling layer to map the low resolution feature to enlarged feature map. [19] proposed event-triggered tracking framework, occlusion and drift identification module is used to identify if the drift has occurred. When drift event occurs, the target re-detection module is activated by the event-triggered decision module to recover the target again in short-term tracking. [20] used a lightweight convolutional network of two layers without offline training to extract a set of normalized patches from the target region. The extracted normalized patches are used as filters to integrate a series of adaptive contextual filters surrounding the target to define a set of feature maps in the subsequent frames. [21] used self-similarity in visual tracking, the target image is divided into non-overlapped patches described by the histogram of gradient (HOG) features. Afterwards, a polynomial kernel feature map is constructed to extract the self-similarity information. A linear support vector machine is used as a classifier. [22] proposed a particle filter framework to handle the appearance changes. The framework used online Fisher discrimination boosting feature selection mechanism to enhance the discriminative capability between the target and background. [23] used a patch based tracker which adaptively integrates the kernel correlation filters with multiple effective features. The template patch is trained by kernel correlation filtering and particle filter framework and adaptively set the weight of each patch for each particle in a particle filtering framework. [24] proposed a regularized correlation filter (CF) based tracking to capture the long-term spatio-temporally nonlocal superpixel appearance information to regularize the CF learning. [25] proposed a boolean map based representation that exploits connectivity cues for visual tracking. The appearance model is described histogram of oriented gradients and raw color features. Boolean maps form together a target representation that can be approximated by an explicit feature map of the intersection kernel, which is fed into a logistic regression classifier. In [26], proposed a high-dimensional multi-scale spatio-colour image feature vector to represent the target object. Afterwards, this feature vector is randomly projected onto a low-dimensional feature space. A feature selection technique is used to design an adaptive appearance model. In [27], an appearance model was developed based on features extracted from a multiscale image feature space with data-independent basis. A non-adaptive random projections is used along with a sparse measurement matrix to extract the features of the appearance model. In most scenarios where drift and significant appearance changes occurs, the above trackers cannot recover the object of interest.

Composite trackers are trackers that combine multiple trackers to track objects. The co-tracking algorithm in [28] used a support vector machine classifier to train with multiple different features and combined their results. The MEEM algorithm [3] used multiple trackers to memorize their past states, so that the tracker can ignore false positive. The unifying algorithm [29] used the relation among individual trackers by measuring the consistency of each tracker between two successive frames and the pair-wise correlation among different trackers. Kown et $al.$[30] decomposed the appearance model into multiple observation models and motion models (VTD) and exploited the results in a unifying tracker within a Bayesian framework. In [31], a Struck tracker [32] was used based on three different feature descriptors; Haar-like features to represent texture information of a target object, color histograms to consider the local color distribution of the target object, and illumination invariant feature. In [33], a refined trajectory of an object is obtained by combining the trajectories of other conventional trackers in a benchmark. [34] used a tracker space, with multiple trackers, and adaptively sampled to run one at a time. Adaptive NormalHedge algorithm [35] proposed an adaptive framework based on a decision-theoretic online learning algorithm called NormalHedge. Adaptive NormalHedge used a set of weighted experts to predict the state of the target and overcomes the fixed percentage factor that is used in the standard NormalHedge. The HDT tracker [36] took feature maps from different CNN layers and used the parameter-free Hedge algorithm [37] to hedge multiple CNN-based trackers into a strong tracker. In [38], the authors proposed a CNN-based tracker to hedge deep features from different layers in the network. The correlation filter is applied to each feature maps from different layers to build up weak trackers which can be hedged into a strong tracker.

Generally, composite trackers will have a pre-set of trackers that can handle different scenarios but they cannot be extended or generalized. Therefore, their ability to handle challenging aerial videos with frequent occlusions and pose changes depends on the individual trackers performance. In most cases, this combination of challenges present in aerial videos causes tracking drift even in composite trackers which lack a mechanism to recover and correct the objects trajectories.

In this chapter, we present an effective tracking framework that has the ability to track objects in challenging aerial videos. The proposed framework has the following novelties:

1. The framework contains two built-in trackers (MDNet [15] and CCOT [13]) and optional user tracker plugins that can be used by the user to include additional trackers in the framework to handle specific scenarios. This allows more flexibility and generalizes the framework to be efficient in different applications.

2. A new mechanism, called the *reporter*, that intervenes when there is a tracking drift and correct the object trajectory.

3. A new metric was developed, the virtual vector shown in Figure 2.7 to be combined with trajectory analysis to calculate a more accurate robustness score.

The rest of this chapter is organized as follows. Section II details the proposed framework. In section III, we present the experimental results obtained on two UAV data-sets and compare

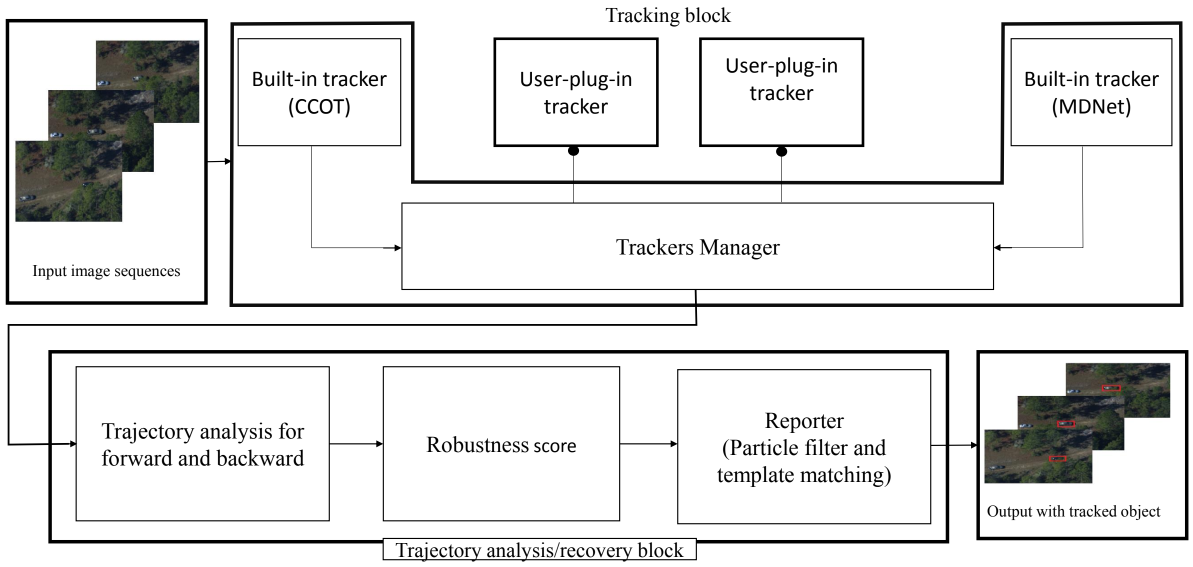


Figure 2.1: Composite framework with the user-plugin trackers and reporter.

them with current state-of-the-art relevant tracking algorithms. Finally conclusions and future work are provided in section IV.

2.3 Proposed framework

In this section, we present the proposed framework as shown in Figure 2.1. The framework mainly consists of two blocks: 1) the tracking block and 2) the trajectory analysis and recovery block. In the first block, a trackers manager controls, manage, and compile the results from the different built-in and user-plugin trackers, more explanation in next sub-section. The second block consists of three steps: 1) the trajectory analysis of forward/backward tracklets, 2) a robustness score calculation, and 3) a reporter mechanism. Figure 2.1 shows the block diagram of the proposed framework. Initially the target is selected then, the framework is initialized . The trajectory analysis and recovery block will be explained in sub-section B. Each tracker in the framework is executed from frame$_{t-n}$ to frame$_t$ to get the forward trajectory and then, from frame$_t$ to frame$_{t-n}$to get the backward trajectory.

2.3.1 The tracking block

Tracklet is a trajectory within a short period of time, it will be used throughout this chapter. The input to this block is the video frames that contain the initial location of the target. The first tracker gives the forward trajectory-1 and backward trajectory-1, while the second tracker gives the forward trajectory-2 and backward trajectory-2. Both trackers work simultaneously to track objects. The optional user trackers plugins are added by user to include additional trackers to handle different scenarios. The tracking block outputs the location trajectories of the target overtime which will be delivered to the second block (trajectory analysis/recovery block) through the trackers manager.

2.3.2 The trajectory analysis and recovery block

The second block receives the trackers results from the trackers manager and process them through a trajectory analysis robustness score, and finally the reporter.

2.3.2.1 Trajectory analysis and robustness for forward and backward

The trajectory is the set of positions of the center of the bounding boxes through the tracking process. Suppose we have a set of frames. We will denote the first frame by frame$_{t-n}$ and the last frame by frame$_t$; where n is any number of frames. Suppose we do not have user-defined trackers, therefore, the composite framework consists of the first trackers that is CCOT and the second one is MD-Net. The framework is executed in two directions, forward trajectory for both trackers; from frame$_{t-n}$ to frame$_t$ ($T_{t\text{-}1f}$ and $T_{t\text{-}2f}$). The outcome of this execution is two forward trajectories. Another execution at the same time is made in reverse direction from frame$_t$ to frame$_{t\text{-}n}$(T$_{t\text{-}1b}$ and T$_{t\text{-}2b}$); the outcome of this execution is two backward trajectories. Figure 2.7 shows the forward and backward trajectories, when they are cyclic, the robustness score will be very close to 1 and that indicates the current tracking result is very accurate and when they are non-cyclic the robustness score has a large value. To the end, we have two forward trajectories and two backward trajectories; in other words, a pair of forward trajectories and a pair of backward trajectories; the first pair of trajectories is from CCOT tracker and the second pair trajectories is from MD-Net tracker. The robustness score is measured for each pair (forward and backward) of trajectories; the trajectories with the highest robustness score are selected as the final trajectories. Consequently, the forward trajectory is the best choice to advance the tracking process within our adaptive framework.

In the trajectory analysis and robustness score, we use the geometric similarity, the cyclic weight, the appearance similarity, and the cosine similarity between virtual vectors. Virtual vectors are developed to calculate the angle between forward and backward trajectories through virtual vectors starting from the ending position and ending at the starting position as shown in Figure 2.7. We develop the virtual vector measure to get more accurate results in terms of robustness score. In Figure 2.7 we assume there are two virtual vectors starting the end of the forward trajectory result and ending where the initial position of the bounding box is located and the location of the object after backward analysis. Consequently, an angle between the two virtual vectors is called θ. A small θ indicates that, the initial location of the target object and the ending location of the object after backward tracking are very close to each other or might be identical. Thus we employ the cosine similarity to measure the angle between the paired virtual vectors as follows

$$Cos(\theta_{\text{t}}) = \frac{\vec{x}_{\text{t0:t29}} \cdot \vec{x}_{\text{t29:t0}}}{\|\vec{x}_{\text{t0:29}}\| \|\vec{x}_{\text{t29:t0}}\|} \tag{2.1}$$

Suppose we have a video sequence of 30 frames, let $\vec{x}_{\text{t}}$ denotes the bounding box location at frame t, which is estimated by the built-in tracker-1 in the forward direction. The forward trajectory from t_0 to t_{29} can be described as follows

$$\vec{x}_{\text{t}_0:\text{t}_{29}} = \{\vec{x}_{\text{t}_0}, \vec{x}_{\text{t}_1}, ..., \vec{x}_{\text{t}_{29}}\} \tag{2.2}$$

where $\vec{x}_{\text{t}_0}$ is the bounding box position at the initial frame, $\vec{x}_{\text{t}_{29}}$ is the bounding box at the last frame. Similarly, the built-in tracker is initialized to trace the target object in the backward direction. The backward trajectory can be described as follows

$$\vec{x}_{\text{t}_{29}:\text{t}_0} = \{\vec{x}_{\text{t}_{29}}, \vec{x}_{\text{t}_{28}}, ..., \vec{x}_{\text{t}_0}\} \tag{2.3}$$

Similarly, the built-in tracker-2 is described in the same way. The geometric similarity can be computed from:

$$\lambda = \exp(-\frac{\|\vec{x}_{\text{t}_0:\text{t}_{29}} - \vec{x}_{\text{t}_{29}:\text{t}_0}\|^2}{\|\sigma^2\|}) \tag{2.4}$$

Where, λ is the geometric similarity and σ^2 is an empirically determined value equals to 500. When the difference between the trajectories is very small, the exponential gives a number very close to 1 and vice versa. In the ideal case, the forward and backward trajectories are identical therefore, the geometric similarity λ equals to 1. This equation will be used later to calculate Eq. 2.6.

In Figure 2.7, at cyclic virtual vectors block the blue trajectory is matched or very close to the red trajectory, therefore this trajectory is selected as a valid trajectory and called cyclic. In such a case the geometric similarity is very close to 1. In contrast, at non-cyclic virtual vectors block the blue forward trajectory does not match the red backward trajectory therefore, we discard this trajectory and we call it non-cyclic, because the initial object can not be accessed again from the backward direction. In such a case the geometric similarity decreases. To calculate the cyclic weight, we count the number of mismatched bounding boxes in the forward trajectory with their correspondences in the backward trajectory from the intersection over union (IoU) as follows:

$$\psi = \frac{\Delta(\vec{x}_{\mathrm{t_0:t_{29}}}, \vec{x}_{\mathrm{t_{29}:t_0}})}{\Delta(\vec{x}_{\mathrm{t_0:t_{29}}}) + \Delta(\vec{x}_{\mathrm{t_{29}:t_0}})} \tag{2.5}$$

Where the denumerator $\Delta(\vec{x}_{\mathrm{t_0:t_{29}}})$ and $\Delta(\vec{x}_{\mathrm{t_{29}:t_0}})$ is the area of the bounding boxes overlap in the forward and backward trajectories. The numerator is the area of the bounding boxes union in the forward and backward trajectories. Practically, we do not need to count the number of mismatched frames ν in the whole period; we consider the first four frames in the forward trajectory which corresponds to the last four frames in the backward trajectory. If the ψ is less than 0.33, a mismatch will be declared. Consequently, the forward and backward trajectories form a non-cyclic trajectory. To assign a weight to the cyclic or non-cyclic trajectories, we will use X where X is an arbitrary number to set the cyclic weight bases quit differently. If the number of mismatched frames ν is 0 or 1 within the first four frames in the forward trajectory, the cyclic weight will be 10^5 otherwise it will be 1.

Now assume we have $\vec{x}_{\mathrm{t_0:t_{29}}}$ and $\vec{x}_{\mathrm{t_{29}:t_0}}$ from the composite framework. The first four frames in the forward trajectory will be denoted by $\vec{t}_{\mathrm{i}}$ where $i \in \{0, 1, 2, 3\}$ and its correspondence from the backward trajectory is $\overleftarrow{t}_{\mathrm{i}}$ where $i \in \{29, 28, 27, 26\}$.

Let $P(\overleftarrow{x}_{\mathrm{t}})$ denote the image patch centered at position x at frame t in the backward trajectory and

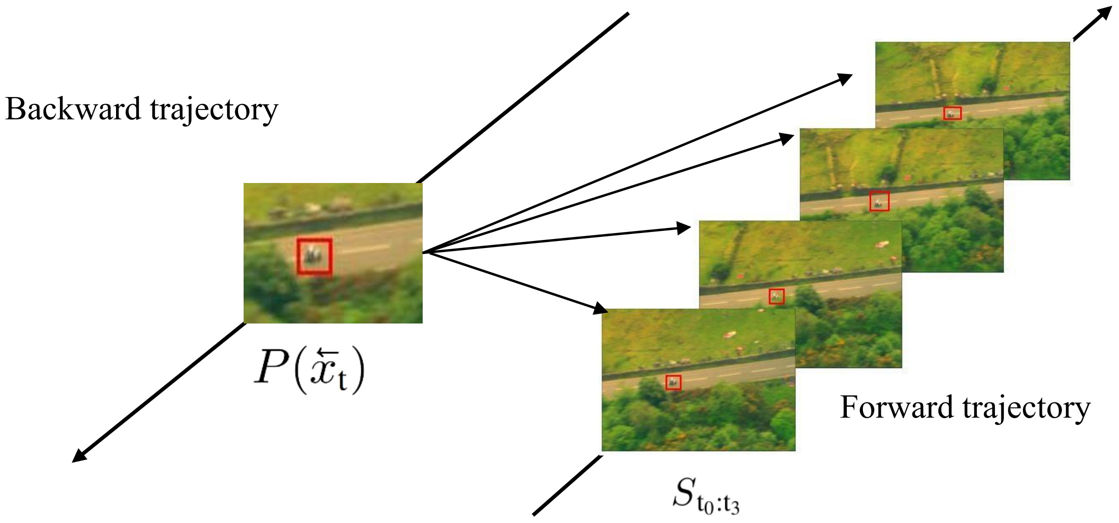


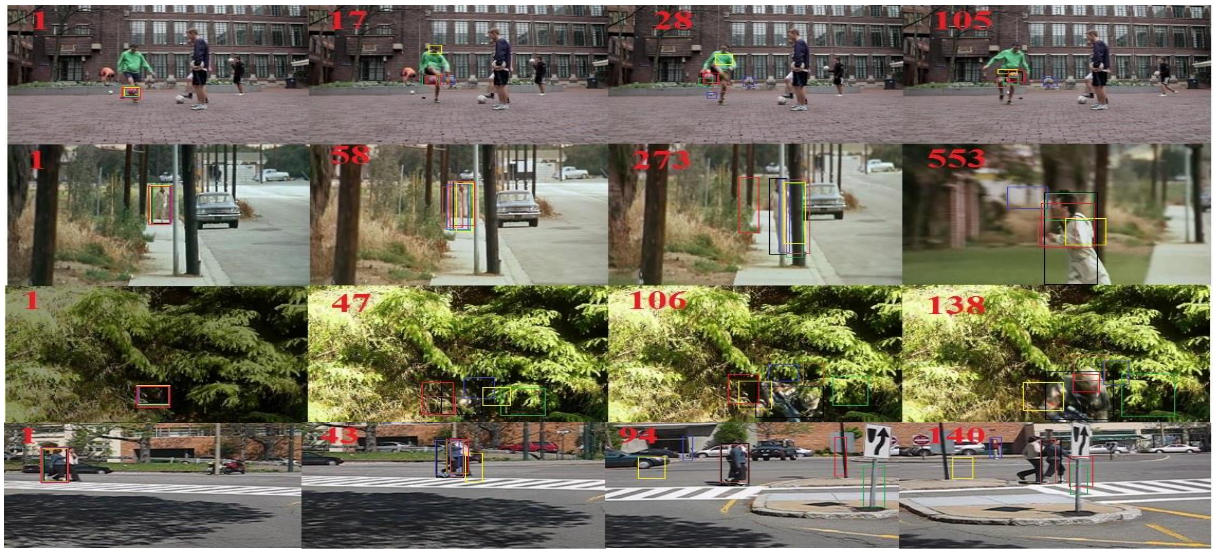
Figure 2.2: Left: the backward trajectory and the centered patch, right: the forward trajectory and the first four frames (set).

$S_{\mathrm{t_0:t_3}}$ denotes the first set of four patches in the forward trajectory as shown in Figure 5.1. The appearance similarity of $P(\overleftarrow{x}_{\mathrm{t}})$ to the set $S_{\mathrm{t_0:t_3}}$ can be calculated from Gaussian kernel between the set and the patch. In general a Gaussian kernel is used to measure the similarity between two vectors or 2-D matrix. The appearance similarity can then be calculated as follows:

$$\phi_{\mathrm{t}} = \exp\left(-\frac{\sum_{Q \in S_{\mathrm{t_0:t_3}}} \|K \cdot (P(\overleftarrow{x}_{\mathrm{t}}) - Q)\|^2}{4\omega h {\sigma^2}_2}\right) \tag{2.6}$$

Where ϕ_{t} is the appearance similarity, ${\sigma^2}_2$ = 900 empirically determined, ω and h are the width and height of the bounding box, respectively. K is a Gaussian weight mask as shown in 2.5, and "$\cdot$" is the element-wise weight multiplication. Small ϕ_{t} indicates high changes in the bounding box appearance or a tracking error.

The larger robustness score is, the more tracking results being accurate. We set the robust-



- MDNet - CCOT - SRDCF - KCF - Ground-Truth - OURS

Figure 2.3: Visual results on VOT2016 data-set for four sequences.

ness score threshold to be 0.65, if one or both trackers scores are greater than the predefined threshold, then the framework will select the highest score. If both trackers scores are less than the predefined threshold, then the framework will activate the reporter mechanism.

When the forward and backward trajectories are identical the similarity becomes 1. If the trajectories are not identical the similarity decreases. Finally, the robustness score for the composite framework can be calculated from :

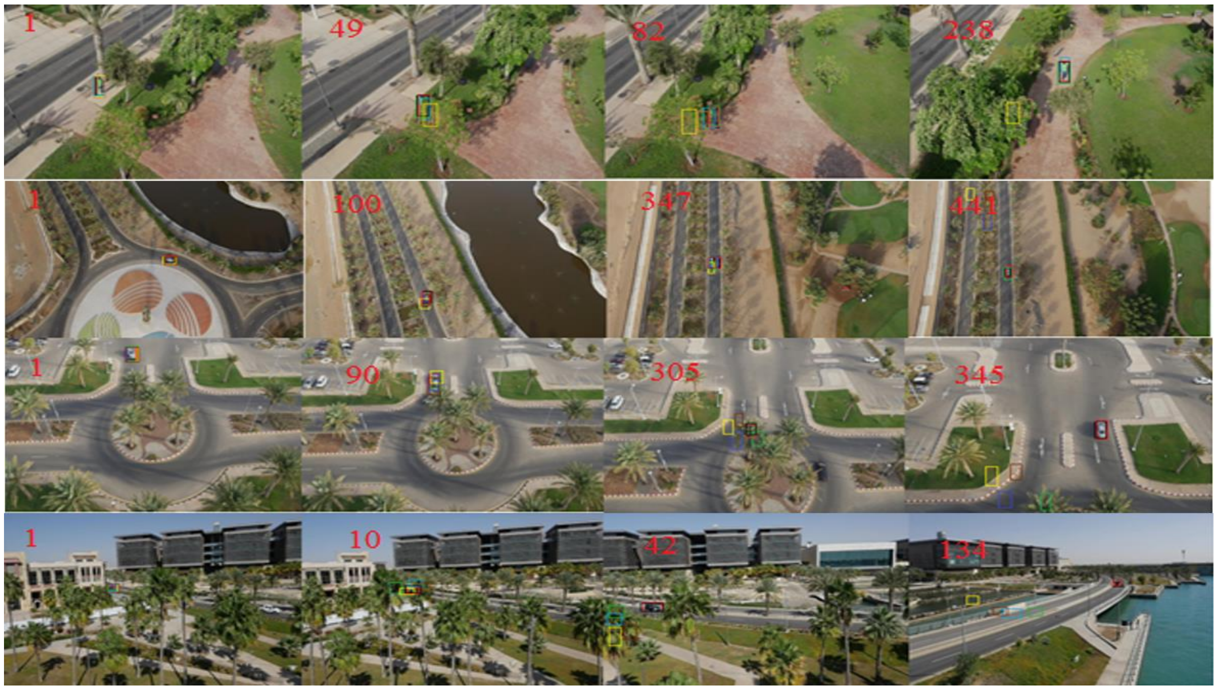
$$\mu_1 = X\left(\sum_{t=t0}^{29} \lambda_{\mathrm{t}} \cdot \phi_{\mathrm{t}} \cdot \cos\theta_{\mathrm{t}}\right) \tag{2.7}$$

Similarly for the built-in tracker-2 μ_2. The normalized score is required to compare both scores to each other. the normalized robustness score will be calculated as follows

$$\hat{\mu}_1 = \frac{\mu_1}{\mu_1 + \mu_2} \tag{2.8}$$

Algorithm 1: Reporter mechanism algorithm

```
    Input : $\mu$: robustness score, $n$: number of particles.
    Initialization : initialize particle filter with $n$ particles.
    Precondition : If $\hat{\mu}_1$ & $\hat{\mu}_2 ... < 0.65$ goto :1 else get the score from Eq. 2.8 and 2.9.
    Output : Recover the lost location of the target object.
1:  foreach particle in next frame do:
2:  create bounding boxes around the $n$ particles
3:  update $n$ using linear motion model
4:  Measure the similarity among the object in the previous frame and the object in the next
    frame using the template matching.
5:  If the matching score $< 0.50$, then the object still occluded/lost, go to 1 if no go to 6
6:  The highest score with the particle which associated the bounding box is the most likely
    the lost location.
7:  The recovered location and its bounding box are fed into the composite tracker
8:  end
```



- MDNet - CCOT - SRDCF - KCF - Ground-Truth - OURS

Figure 2.4: Visual results on UAV123 data-set for four sequences.

$$\hat{\mu}_2 = \frac{\mu_2}{\mu_1 + \mu_2} \tag{2.9}$$

Maximum score represents the best trajectory, as it tells how similar the forward and backward trajectories are to each other. The more similar the trajectories are to each other, the higher the value of the robustness score.

2.3.2.2 The reporter

The proposed reporter mechanism as shown in algorithm 1 consists of the particle filter and the template matching. It only works when the robustness score is less than the predefined threshold which is 0.65 in our framework through this chapter. When the robustness score is less than the threshold, the particle filter will be initialized by 300 particles around the center

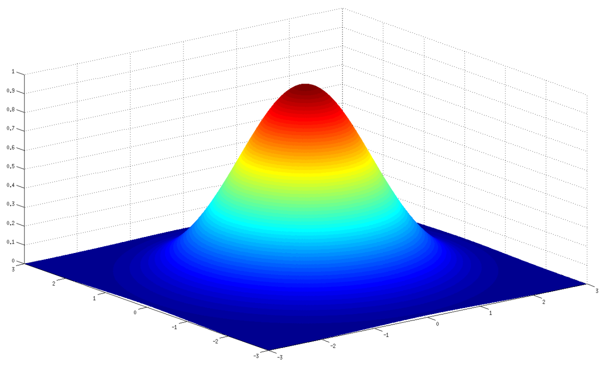


Figure 2.5: Normalized kernel Gaussian mask (K).

of the object, each particle is associated with five states as shown in Eq. 2.10. Therefore, the particle filter updates the states using the linear motion model of the previous object (which the object in frame$_{\text{t-n}}$) to the future states.

$$States = (x_{\text{t}}, y_{\text{t}}, s_{\text{t}}, \alpha_{\text{t}}, \theta_{\text{t}}) \tag{2.10}$$

where x_{t}, y_{t}, s_{t}, α_{t}, θ_{t} are x, y translations, scale, aspect ratio, and in-plane rotation angle respectively. At each particle, a bounding box is created around the location of particle; the size of the bounding box is 36×36 pixels since we work on very tiny objects in UAV. Afterward, template matching is used to calculate the similarity among the object in the previous frame$_{\text{t-n}}$ and all bounding boxes where all particles are located. The higher score of template matching is, the more likely the location of object is correct in frame$_{\text{t}}$. Now the recovered location of the target object will be the input to composite trackers. Finally the framework takes the input images and calculates the forward/backward trajectories by the tracking block which has two trackers. The analysis of these trajectories is done in trajectory analysis and recovery block to give the final result which is the location of the object of interest.

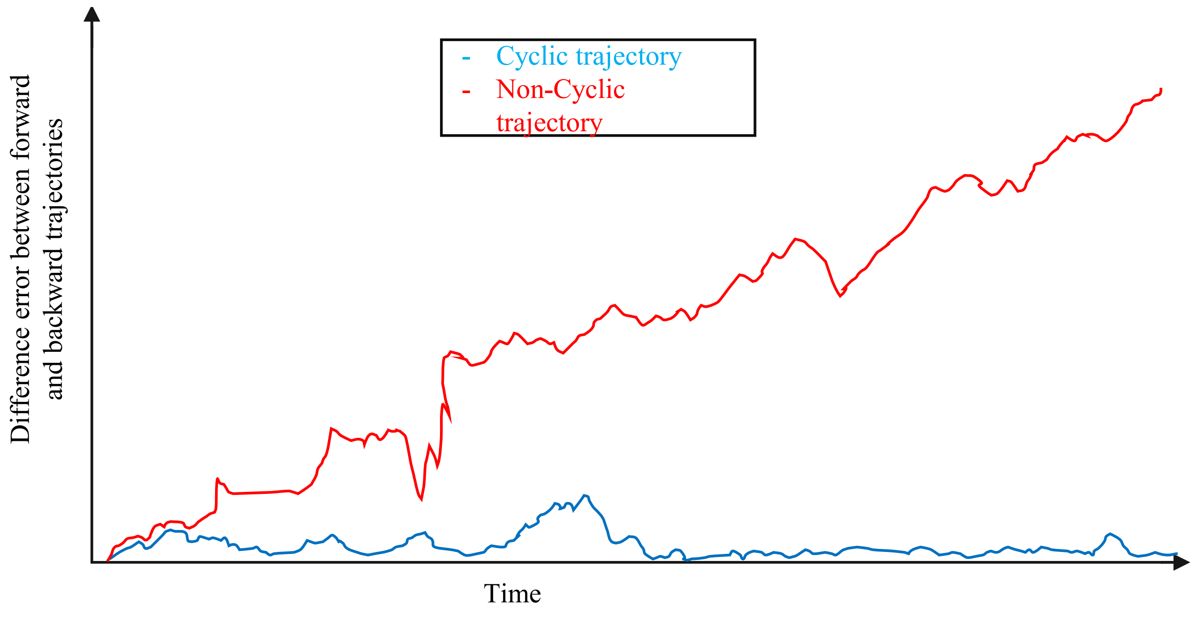


Figure 2.6: Cyclic and non-cyclic trajectories

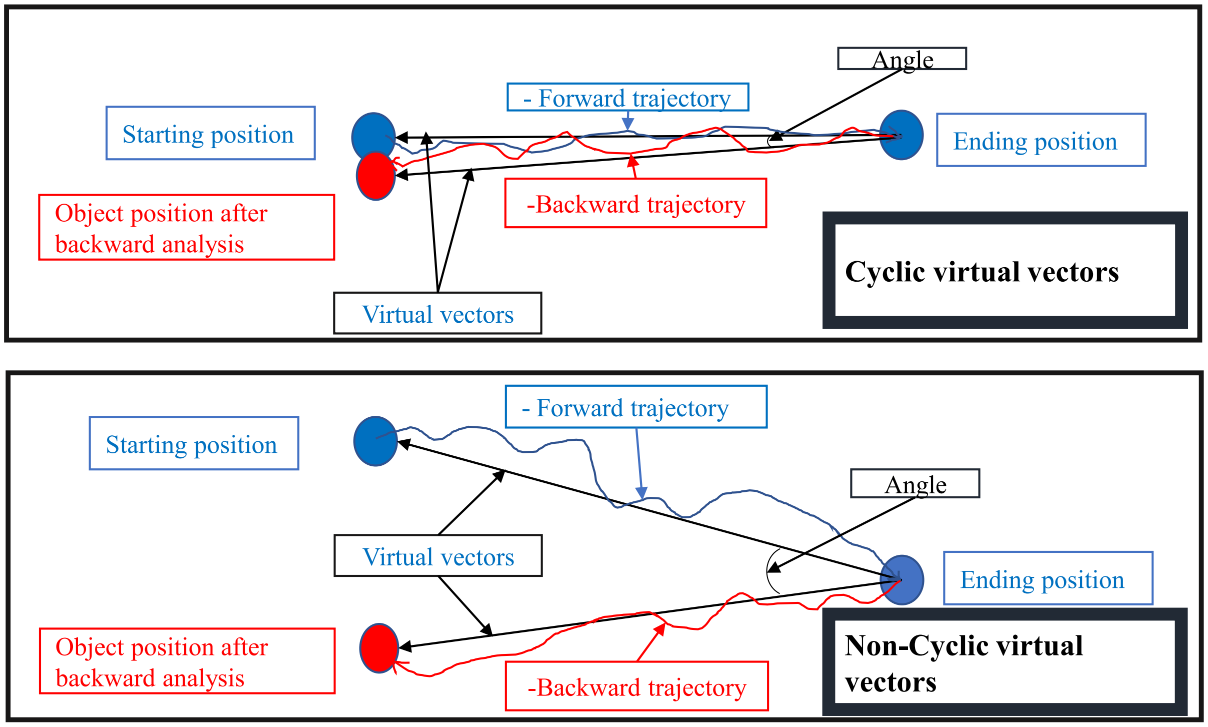


Figure 2.7: Virtual vectors representation.

Table 2.1: Overlap rate and the average time in each frame against state-of-the-art trackers on VIVID-EGTest and VOT data-sets

Trackers	Sequences													
	01		02		03		04		05		VOT2016Road		Overall	
	OVR	Time(s)	OVR	Time(s)	OVR	Time(s)	OVR	Time(s)	OVR	Time(s)	OVR	Time(s)	OVR	Time(s)
MDNet	0.71	1	0.69	1.1	0.82	1.01	**0.81**	1	**0.84**	1.07	**0.78**	1.09	0.78	1.04
CCOT	0.79	0.05	**0.74**	0.06	**0.83**	0.05	0.77	0.08	0.83	0.07	0.77	0.089	**0.79**	0.07
SRDCF	**0.80**	0.2	0.62	0.2	0.79	0.8	0.73	0.3	0.79	0.7	**0.78**	0.88	0.75	0.51
KCF	0.72	0.005	0.65	0.002	0.77	0.008	0.76	0.005	0.76	0.007	0.75	0.008	0.73	0.006
MTA	0.73	0.08	0.69	0.10	0.83	0.09	0.73	0.07	0.82	0.12	0.80	0.08	0.77	0.09
OURS	**0.87**	0.25	**0.76**	0.22	**0.86**	0.17	**0.88**	1.23	**0.89**	1.20	**0.80**	1.24	**0.84**	0.72

2.4 Experimental Results

In this section we will provide the parameters that were used in our experiment to make this approach reproducible with the same results. Also a qualitative discussion is provided by the end of this section. Each centered image of the target object is re-sized to be a 36 $\times$ 36 patch. To initialize the particle filter, 300 particles were used. The frame numbers n is set to $n = 30$ The implementation was performed using MATLAB-2017b, a computer with Core I7 CPU, 2.1 GHz processor with TITAN XP GPU, 64-GB RAM and no code optimization. We used VOT2016 [39], Vivid [40] and UAV123 [41] datasets to evaluate our proposed framework as shown in Figure 2.11. The overlap can be calculated as follows $S = \frac{r_t \cap r_a}{r_t \cup r_a}$, where r is the bounding box,

Table 2.2: Comparison with the state-of-the-art trackers on VOT2015.

Tracker	A	R	EAO	FPS
MDNet	0.60	0.69	0.38	1
DeepSRDCF	0.56	1.05	0.32	< 1
EBT	0.47	1.02	0.31	4.4
SRDCF	0.56	1.24	0.2	5
BACF	0.59	1.56	–	35
EAST	0.57	1.03	0.34	159
Staple	0.57	1.39	0.30	80
SamFC	0.55	1.58	0.29	86
Ours	0.612	0.67	0.39	0.72

Table 2.3: Comparison with the state-of-the-art trackers on VOT2016.

Tracker	A	R	EAO	FPS
ECOhc	0.54	1.19	0.3221	60
Staple	0.54	1.42	0.2952	80
STAPLE+	0.55	1.31	0.2862	> 25
SiamRN	0.55	1.36	0.2766	> 25
GCF	0.51	1.57	0.2179	> 25
Ours	0.55	1.15	0.3308	0.72

$\cap$, and $\cup$ are the intersection and union of two bounding boxes, respectively.

Based on ground truth data, Figure 2.3 visually compares the tracking results obtained us-

Table 2.4: Comparison with the state-of-the-art trackers on VOT2017.

Tracker	A	R	EAO	FPS
SiamDCF	0.500	0.473	0.249	60
ECOhc	0.494	0.435	0.238	60
CSRDCF++	0.453	0.370	0.229	> 25
SiamFC	0.502	0.585	0.188	86
SAPKLTF	0.482	0.581	0.184	> 25
Staple	0.530	0.688	0.169	> 80
ASMS	0.494	0.623	0.169	> 25
Ours	0.540	0.370	0.250	0.72

Table 2.5: Comparison of state-of-the-art trackers on OTB-50 and OTB-100.

	Tracker	Ours	MDNet	CCOT	LMCF	CFNet	Staple	PTAV	SiamFC	ECOhc
OTB-50	AUC	**0.669**	0.645	0.614	0.533	0.530	0.507	0.581	0.516	0.592
	Prec.	**0.936**	0.890	0.843	0.730	0.702	0.684	0.806	0.692	0.814
OTB-100	AUC	**0.708**	0.678	0.671	0.580	0.568	0.578	0.635	0.582	0.643
	Prec.	**0.930**	0.909	0.898	0.789	0.748	0.784	0.849	0.771	0.856
Speed	FPS	0.86	1	0.3	85	75	80	25	86	60

ing different state-of-the-art trackers against the proposed tracker. It shows that our approach is more robust and handles most of four sequences well although very fast motion in row 1 or occlusion in row 2 except the row 3 where mismatch occurs at frame106 in VOT2016 dataset [39]. In this sequence, the object is diffused with background; in such case the robustness score

Table 2.6: Ablation study of performance evaluation for adding user-plug-in trackers (ECOhc and SiamFC) to the propsed framework.

Proposed Tracker					Dataset				Speed
Built-in	User-plug-in		Components		OTB-50		OTB-100		FPS
Baseline	ECOhc	SiamFC	Reporter	Virtual-vector	AUC	Prec.	AUC	Prec.	-
✓					0.646	0.899	0.680	0.908	0.8926
✓			✓		0.650	0.909	0.688	0.910	0.8906
✓				✓	0.651	0.910	0.687	0.909	0.8920
✓			✓	✓	0.654	0.917	0.693	0.915	0.8898
✓	✓				0.649	0.909	0.689	0.913	0.8759
✓	✓		✓		0.653	0.912	0.695	0.917	0.8739
✓	✓			✓	0.655	0.923	0.693	0.916	0.8753
✓	✓		✓	✓	0.660	0.931	0.708	0.923	0.8730
✓	✓	✓			0.654	0.924	0.692	0.915	0.8590
✓	✓	✓	✓		0.657	0.928	0.698	0.922	0.8571
✓	✓	✓		✓	0.656	0.927	0.700	0.924	0.8584
✓	✓	✓	✓	✓	0.669	0.936	0.708	0.930	0.8563

declares that the forward and backward trajectories are not similar or the object has been lost. The reporter starts to work and the particle filter will create 300 particles and its corresponding patches. Then the reporter mechanism successes to recover the object in the next frames. whereas many existing trackers have errors propagated. Also in row 4, frame #10 our approach reports that the robustness score is 0.22 which is less than the predefined threshold; In this case

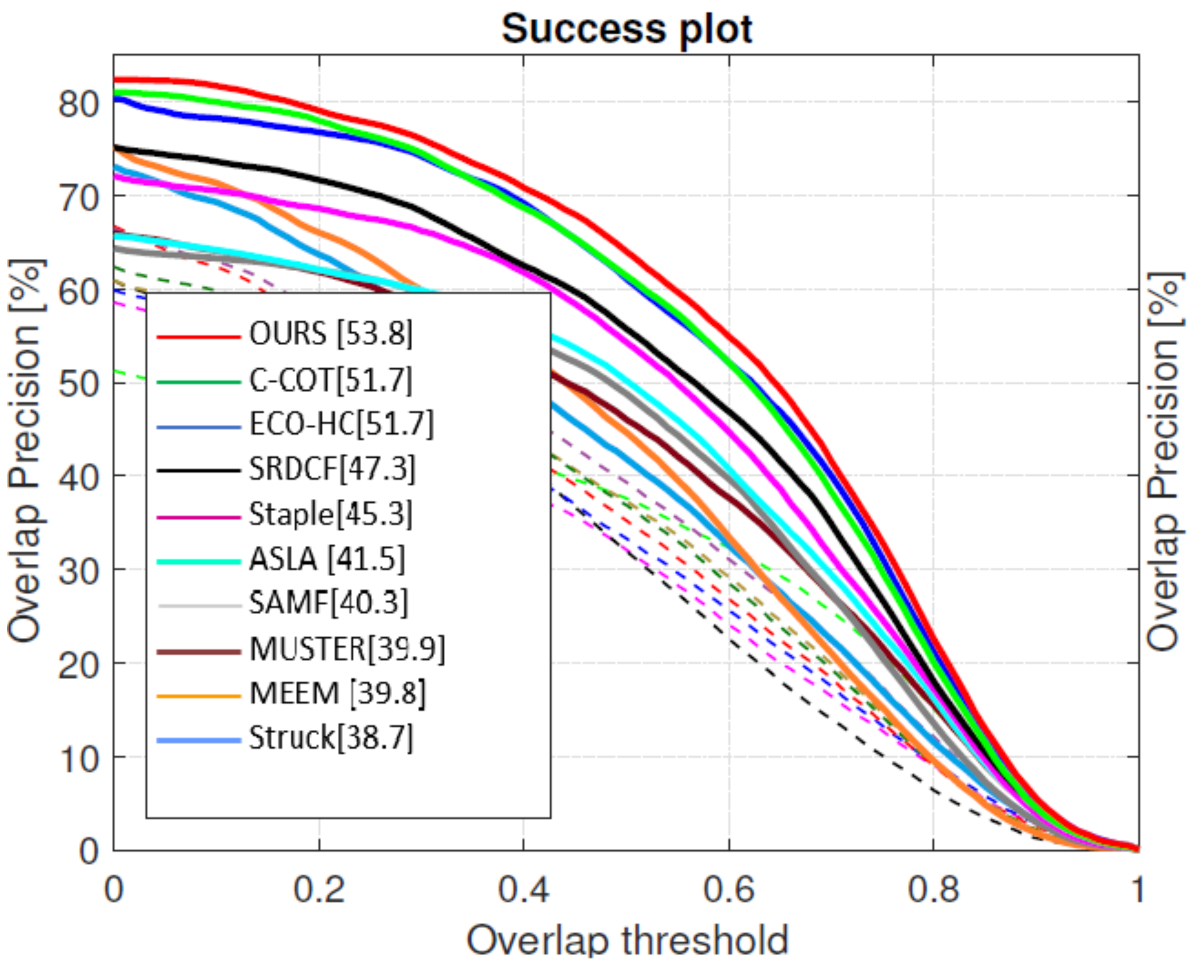


Figure 2.8: Success plot on UAV123 for top 10 trackers. Legend shows AUC.

the reporter starts to work and it will create the 300-particle and unfortunately none of them is overlapped ,therefore, the framework reports that, the object is lost. we sample these particles randomly from Gaussian distribution however, this case is very rare since the particles are randomly distributed around the object.

Table 2.1 summarizes the results in terms of overlap rates and the time spent in each frame for the five sequences in VIVId-EGTest dataset and VOT-2016 road sequence. It confirms that our approach is able to properly track the object with lowest tracking errors.

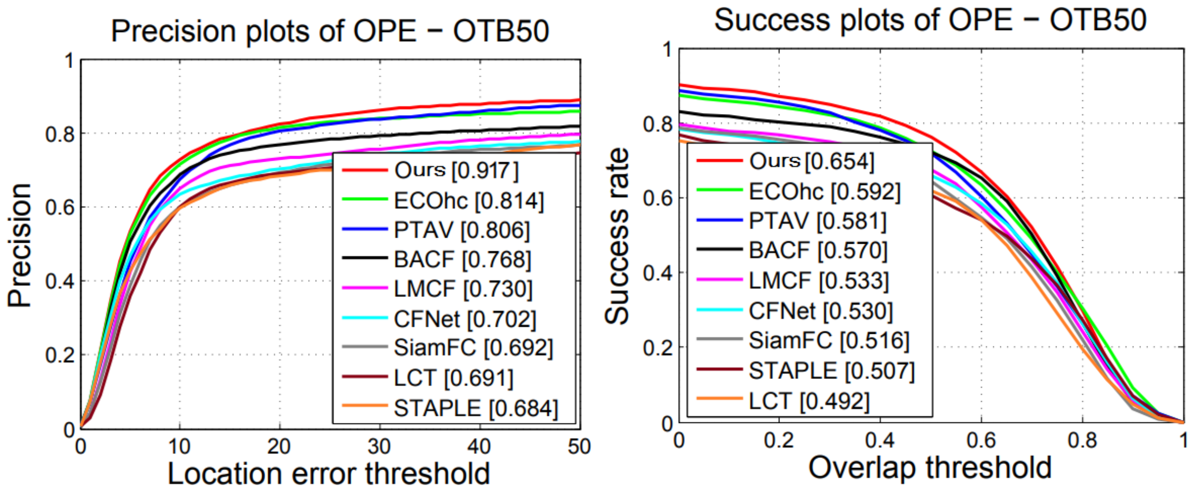


Figure 2.9: Precision and success plots on OTB-50 benchmark. All curves and numbers are generated from OTB toolkit.

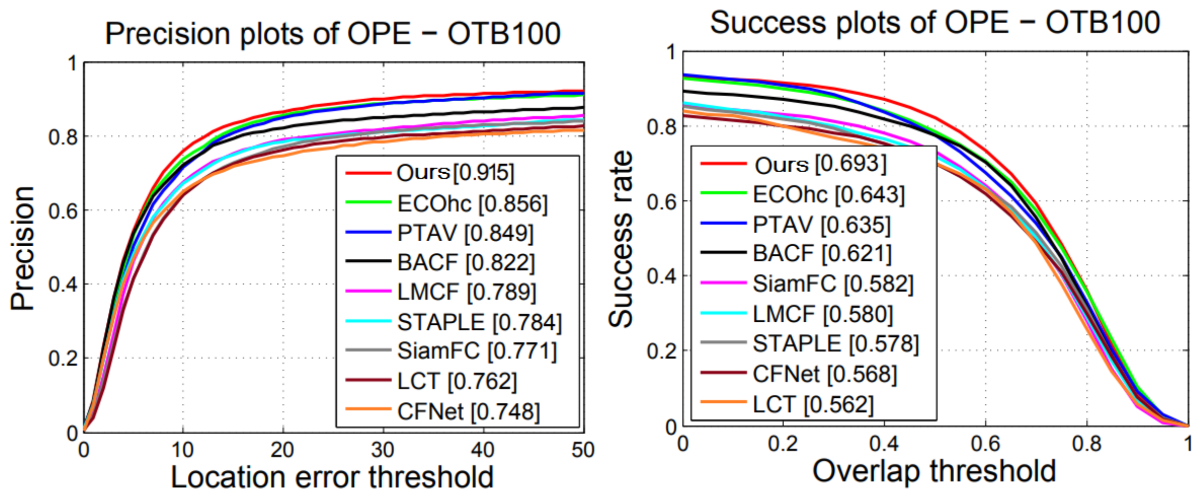
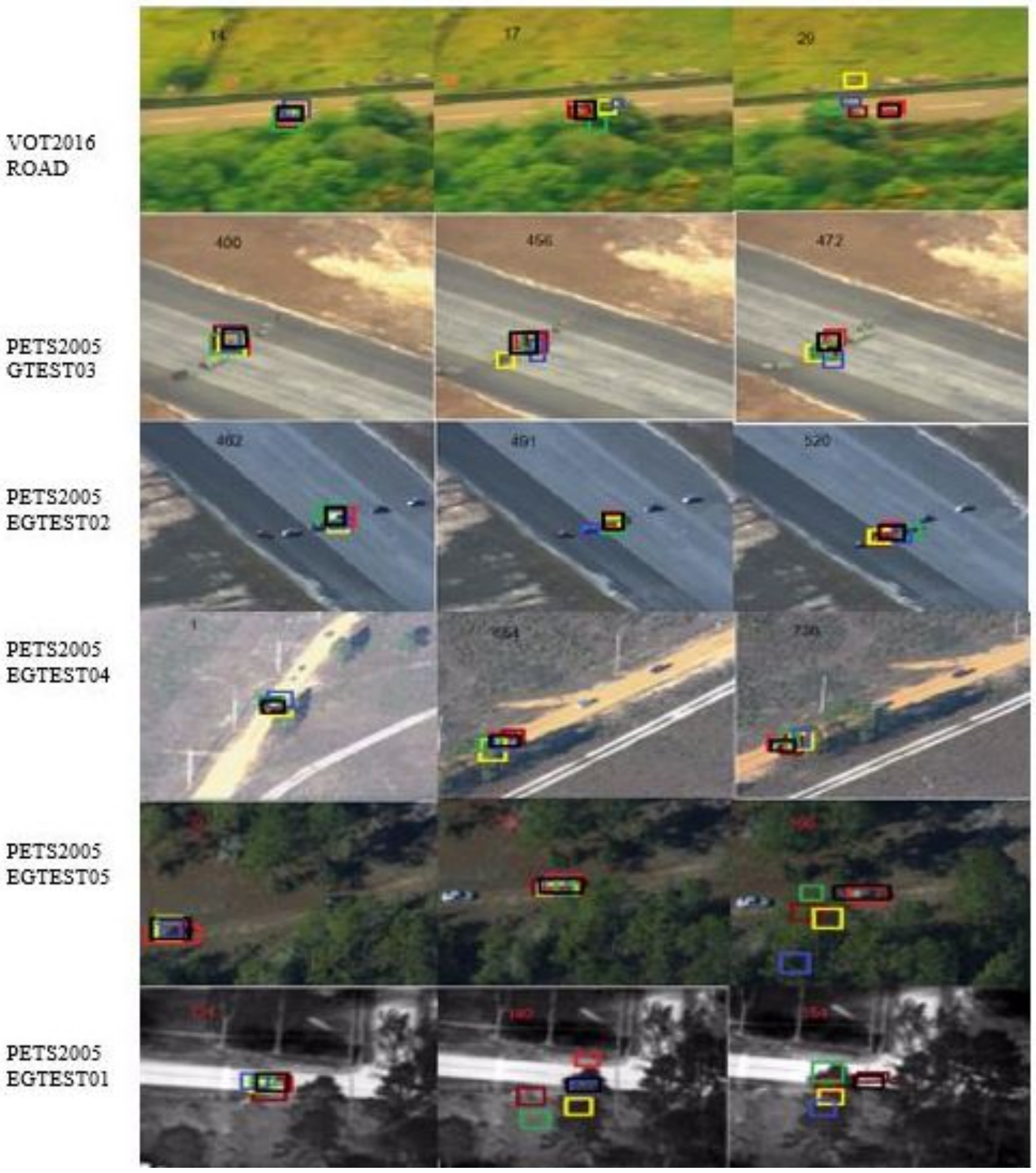


Figure 2.10: Precision and success plots on OTB-100 benchmark. All curves and numbers are generated from OTB toolkit.



- MDNet - CCOT - SRDCF - KCF - Ground-Truth - OURS

Figure 2.11: Visual results on two data-sets VOT2016-Road and VIVId-EGTest.

Figure 2.4 shows the visual results on UAV123 [41] data-set, we ran and evaluated 9 trackers in addition to ours on UAV123 data-set using success plot [33], and calculated the percentage of frames that is within a threshold with an intersection-over-union (IOU). Figure 2.8 ranks trackers acoording to their area-under-curve (AUC) score. CCOT runs at (0.30 FPS) and its AUC is 51.7%. Our tracker runs at (0.86 FPS) with an AUC score of 53.8% which is outperforming CCOT by 2.1%. At the first row all trackers fail to track the object of interest due to occlusion except CCOT tracker and ours. Moreover, when the object undergoes full occlusion such as row #3 all trackers drift off. However, our tracker still can track the object, in this case the reporter mechanism works and particle filter starts to propagate the particles to find the most similar patch to the object of interest. Figure 2.11 shows the results on VIVID-EGTest data-set and Vot2016-road sequenc. The first row shows that the object undergoes full occlusion in frame #17, all trackers fail to track the object however, our tracker can find the object after occlusion. We ran our tracker on VIDI-EGTest data-set, row #2 to row #6 show different challenges such as occlusion (row #2), very tiny objects (row #3), sudden discontinuity (jump forward) (row #4), illumination changes (row #5) and frequent occlusion by trees(row #6). Our tracker can handle all these scenarios compared to the other tracker.

Extensive experiments are conducted to evaluate our tracker against the state-of-the-art trackers on OTB-50, OTB-100, VOT2015, VOT2016 and VOT2017 benchmarks. All experiments in this section were done using only the built-in trackers except the experiment in Table 5.7. Table 5.2 shows the performance of our tracker against eight state-of-the-art trackers in terms of accuracy (A), robustness score(R) and expected average overlap (EAO). The first four trackers in Table 5.2 are non-real-time while other trackers are working in real-time. In terms of accuracy the proposed framework is outperforming other trackers especially MDNet by 1.2%. The robustness of the proposed framework is the best compared to all other trackers, the gap between the second best tracker (MDNet) is 2%. Consequently, the expected average overlap for the proposed framework has increased compared to MDNet by 1%. Table 5.3 and Table 4.1 show the performance of the proposed framework against five and eight state-of-the-art trackers respectively. The proposed framework outperforms all other trackers. Figure 2.5 and Figure 2.10 show the precision and area-under-curve (AUC) on OTB-50 and OTB-100 respectively, all curves are generated from OTB toolkit. Table 2.5 shows that the proposed framework outperforms all listed trackers on OTB-50 and OTB100.

The proposed framework outperforms other methods because it relies on the trajectory analysis from each tracker. The framework chooses the best trajectory pair (forward and backward) based on the highest score of robustness. On top of that, in case all trackers drift off, the framework detects that the object is lost and the reporter mechanism starts to work by creating 300 particles to find the most similar patch to the object of interest. The performance of the proposed framework on OTB benchmark better than VOT benchmark by 10% since the VOT has very challenging sequences. Figure 2.3 shows at the first row a very challenging sequence, the object (ball) moves very fast also in Road-sequence the object undergoes a full occlusion. However, the proposed tracker outperforms the state-of-the-art trackers.

2.4.1 Ablation Study

In this experiment, we show the effect of adding more trackers (user-plug-in) to the framework. In addition, we show the effect of framework variation components such as the reporter and virtual vector. Table 5.7 lists the variation components of the proposed framework. Baseline means using only the framework with built-in trackers without the reporter and virtual vector. The first row shows that, using only the baseline hardly improves the performance. The second row shows that, adding the reporter to baseline improves the overall performance. This confirms the importance of the reporter. In the third row, adding virtual vector to baseline without the reporter improves the performance as reporter with almost the same performance. In the fourth row, adding the reporter and virtual vector to the framework improve the overall performance. Table 5.7 also lists the tested user-plug-in trackers(ECOhc and SiamFC). Obviously, adding user-plug-in trackers with reporter and virtual vector significantly improve the overall performance.

2.5 Conclusion

In this chapter, a composite framework for unmanned vehicle tracking is presented. The composite framework consists of two trackers with trajectory analysis and virtual vectors. The composite framework uses the forward and backward trajectories. A new mechanism called reporter is used to make the tracker more robust. The reporter uses the robustness score and particle filter to decide which trajectory will be selected from the forward pairs.

Extensive experiments were conducted on OTB-50, OTB-100, UAV123, VOT2015, VOT2016 and VOT2017. The experiments have shown that, adding user-plugins, reporter and virtual vector to the robustness score increased the robustness of the proposed framework. Future work includes a deep convolutionl reporter within the composite framework and using the moving horizon estimation instead of particle filter.

References

[1] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *Acm computing surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[2] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, “Incremental learning for robust visual tracking,” *International Journal of Computer Vision*, vol. 77, no. 1, pp. 125–141, 2008.

[3] J. Zhang, S. Ma, and S. Sclaroff, “Meem: robust tracking via multiple experts using entropy minimization,” in *European Conference on Computer Vision*. Springer, 2014, pp. 188–203.

[4] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao, “Multi-store tracker (muster): A cognitive psychology inspired approach to object tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 749–758.

[5] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, “Accurate scale estimation for robust visual tracking,” in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[6] Q. Wang, F. Chen, W. Xu, and M.-H. Yang, “Object tracking via partial least squares analysis,” *IEEE Transactions on Image Processing*, vol. 21, no. 10, pp. 4454–4465, 2012.

[7] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[8] M. J. Black and A. D. Jepson, “Eigentracking: Robust matching and tracking of articulated objects using a view-based representation,” *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[9] S. He, Q. Yang, R. W. Lau, J. Wang, and M.-H. Yang, “Visual tracking via locality sensitive histograms,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2427–2434.

[10] L. Sevilla-Lara and E. Learned-Miller, “Distribution fields for tracking,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1910–1917.

[11] O. Zoidi, A. Tefas, and I. Pitas, “Visual object tracking based on local steering kernels and color histograms,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 5, pp. 870–882, 2013.

[12] S. Avidan, “Support vector tracking,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 8, pp. 1064–1072, 2004.

[13] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, “Beyond correlation filters: Learning continuous convolution operators for visual tracking,” in *European Conference on Computer Vision*. Springer, 2016, pp. 472–488.

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[15] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[16] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, “End-to-end repre-

sentation learning for correlation filter based tracking," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5000–5008.

[17] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," *arXiv preprint arXiv:1802.08817*, 2018.

[18] X. Lu, H. Huo, T. Fang, and H. Zhang, "Learning deconvolutional network for object tracking," *IEEE Access*, vol. 6, pp. 18 032–18 041, 2018.

[19] M. Guan, C. Wen, M. Shan, C.-L. Ng, and Y. Zou, "Real-time event-triggered object tracking in the presence of model drift and occlusion," *IEEE Access*, 2018.

[20] K. Zhang, Q. Liu, Y. Wu, and M.-H. Yang, "Robust visual tracking via convolutional networks without training," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1779–1792, 2016.

[21] H. Song, Y. Zheng, and K. Zhang, "Robust visual tracking via self-similarity learning," *Electronics Letters*, vol. 53, no. 1, pp. 20–22, 2016.

[22] J. Yang, K. Zhang, and Q. Liu, "Robust object tracking by online fisher discrimination boosting feature selection," *Computer Vision and Image Understanding*, vol. 153, pp. 100–108, 2016.

[23] W. Chen, K. Zhang, and Q. Liu, "Robust visual tracking via patch based kernel correlation filters with adaptive multiple feature ensemble," *Neurocomputing*, vol. 214, pp. 607–617, 2016.

[24] K. Zhang, X. Li, H. Song, Q. Liu, and W. Lian, "Visual tracking using spatio-temporally nonlocally regularized correlation filter," *Pattern Recognition*, 2018.

[25] K. Zhang, Q. Liu, J. Yang, and M.-H. Yang, “Visual tracking via boolean map representations,” *Pattern Recognition*, vol. 81, pp. 147–160, 2018.

[26] H. Song, “Robust visual tracking via online informative feature selection,” *Electronics Letters*, vol. 50, no. 25, pp. 1931–1933, 2014.

[27] K. Zhang, L. Zhang, and M.-H. Yang, “Fast compressive tracking,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 1, pp. 1–1, 2014.

[28] F. Tang, S. Brennan, Q. Zhao, and H. Tao, “Co-tracking using semi-supervised support vector machines,” in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8.

[29] Y. Gao, R. Ji, L. Zhang, and A. Hauptmann, “Symbiotic tracker ensemble toward a unified tracking framework,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 7, pp. 1122–1131, 2014.

[30] J. Kwon and K. M. Lee, “Visual tracking decomposition,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1269–1276.

[31] D.-Y. Lee, J.-Y. Sim, and C.-S. Kim, “Multihypothesis trajectory analysis for robust visual tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5088–5096.

[32] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[33] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[34] J. Kwon and K. M. Lee, “Tracking by sampling and integratingmultiple trackers,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1428–1441, 2014.

[35] S. Zhang, H. Zhou, H. Yao, Y. Zhang, K. Wang, and J. Zhang, “Adaptive normalhedge for robust visual tracking,” *Signal Processing*, vol. 110, pp. 132–142, 2015.

[36] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang, “Hedged deep tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4303–4311.

[37] K. Chaudhuri, Y. Freund, and D. J. Hsu, “A parameter-free hedging algorithm,” in *Advances in neural information processing systems*, 2009, pp. 297–305.

[38] Y. Qi, S. Zhang, L. Qin, Q. Huang, H. Yao, J. Lim, and M.-H. Yang, “Hedging deep features for visual tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[39] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin Zajc, T. Vojir, G. Häger, A. Lukežič, and G. Fernandez, “The visual object tracking vot2016 challenge results,” Springer, Oct 2016. [Online]. Available: http://www.springer.com/gp/book/9783319488806

[40] R. Collins, X. Zhou, and S. K. Teh, “An open source tracking testbed and evaluation

web site," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2005)*, vol. 2, 2005, p. 35.

[41] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *European Conference on Computer Vision*. Springer, 2016, pp. 445–461.

Chapter 3

DensSiam: End-to-End Densely-Siamese Network with Self-Attention Model for Object Tracking

3.1 Abstract

Convolutional Siamese neural networks have been recently used to track objects using deep features. Siamese architecture can achieve real time speed, however it is still difficult to find a Siamese architecture that maintains the generalization capability, high accuracy and speed while decreasing the number of shared parameters especially when it is very deep. Furthermore, a conventional Siamese architecture usually processes one local neighborhood at a time, which makes the appearance model local and non-robust to appearance changes.

To overcome these two problems, this chapter proposes DensSiam, a novel convolutional Siamese architecture, which uses the concept of dense layers and connects each dense layer to

all layers in a feed-forward fashion with a similarity-learning function. DensSiam also includes a Self-Attention mechanism to force the network to pay more attention to the non-local features during offline training. Extensive experiments are performed on four tracking benchmarks: OTB2013 and OTB2015 for validation set; and VOT2015, VOT2016 and VOT2017 for testing set. The obtained results show that DensSiam achieves superior results on these benchmarks compared to other current state-of-the-art methods.

3.2 Introduction

Visual object tracking is an important task in many computer vision applications such as image understanding [1], surveillance [2], human-computer interactions [3] and autonomous driving [4]. One of the main challenges in object tracking is how to represent the appearance model in such a way that the model is robust to appearance changes such as motion blur, occlusions, background clutter [5, 6]. Many trackers use handcrafted features such as CACF [7], SRDCF [8], KCF [9] and SAMF [10] which have inferior accuracy and/or robustness compared to deep features.

In recent years, deep convolutional neural networks (CNNs) have shown superior performance in various vision tasks. They also increased the performance of object tracking methods. Many trackers have been developed using the strength of CNN features and significantly improved their accuracy and robustness. Deep trackers include SiamFC [11], CFNet [12], DeepSRDCF [13], HCF [14]. However, these trackers exploit the deep features which were originally designed for object recognition and neither consider the temporal information such as SiamFC [11] nor non-local features such as the rest of the trackers. The key to design a high-performance tracker is to find the best deep architecture that captures the non-local features while maintaining the real-time speed. Non-local features allow the tracker to be well-generalized from one domain to another domain (e.g. from ImageNet videos domain to OTB/VOT videos domain).

In this chapter, we present a novel network design for robust visual object tracking to improve the generalization capability of Siamese architecture. DensSiam network solves these issues and produces a non-local response map. DensSiam has two branches, the target branch that takes the input target patch from the first frame and the search branch that takes later images in the whole sequence. The target branch consists of dense blocks separated by a transition layer, each block has many convolutional layers and each transition layer has a convolutional layer with an average pool layer. Each dense layer is connected to every other layer in a feed-forward fashion. The target response map is fed into the Self-Attention module to calculate response at a position as a weighted sum of the features at all positions. The search branch is the same architecture as the target branch except that it does not have the Self-Attention module since we calculate the similarity function between the target patch with non-local features and the candidate patches. Both target and search branches share the same network parameters across channels extracted from the same dense layers. To the best of our knowledge, this is the first densely Siamese network with a Self-Attention model.

To summarize, the main contributions of this work are three-fold.

- A novel end-to-end deep Densely-Siamese architecture is proposed for object tracking. The new architecture can capture the non-local features which are robust to appearance changes. Additionally, it reduces the number of shared parameters between layers while building up deeper network compared to other existing Siamese-based architectures commonly used in current state-of-the-art trackers.

- An effective response map based on Self-Attention module that boosts the DensSiam tracker performance . The response map has no-local features and captures the semantic information about the target object.

- The proposed architecture tackles the vanishing-gradient problem, leverages feature reuse and improves the generalization capability.

The rest of the chapter is organized as follows. We first introduce related work in Section 7.3. Section 7.4 details the proposed approach. We present the experimental results in Section 7.7. Finally, Section 3.6 concludes the this chapter.

3.3 Related work

There are extensive surveys on visual object tracking in the literatures [15]. Recently, deep features have demonstrated breakthrough accuracies compared to handcrafted features. Deep-SRDCF [13] uses deep features of a pretrained CNN (e.g. VGG [16]) from different layers and integrate them into a correlation filter. Visual object tracking can be modeled as a similarity learning function in an offline training phase. Siamese architecture consists of two branches, the target object branch and the search image branch. Siamese architecture takes the advantage of end-to-end learning. Consequently, the learned function can be evaluated online during tracking.

The pioneering work for object tracking is the SiamFC [11]. SiamFC has two branches, the target branch and the appearance branch, a correlation layer is used to calculate the correlation between the target patch and the candidate patches in the search image. The search image is usually larger than the target patch to calculate the similarities in a single evaluation. CFNet

[12] improved SiamFC by introducing a differentiable correlation filter layer to the target branch to adapt the target model online. DSiam [17] uses a fast transfer motion to update the leaned model online. Significantly improved performance as it captures some information about the object's context.

SINT [18] uses optical flow and formulates the visual object tracking as a verification problem within Siamese architecture, it has a better performance however, the speed dropped down from 86 to 4 frames per second. SA-Siam [19] uses two Siamese networks based on the original Siamese architecture [11]. The first network for semantic information and the other one for appearance model. This architecture significantly improved the tracker performance as it allows the semantic information of the appearance model representation to incorporate into the response map. However these trackers use the features taken directly from CNN which processes the information in a local neighbourhood. Consequently the output features are local and do not have a lot information about the neighbourhood. Thus using convolutional layers alone is not effective to capture the generic appearance model.

3.4 Proposed Approach

We propose effective and efficient deep architecture for visual tracking named Densely-Siamese network (DensSiam). Fig. 3.1 shows the DensSiam architecture of the proposed tracker. The core idea behind this design is that, using densely blocks separated by transition layers to build up Siamese network with Self-Attention model to capture non-local features.

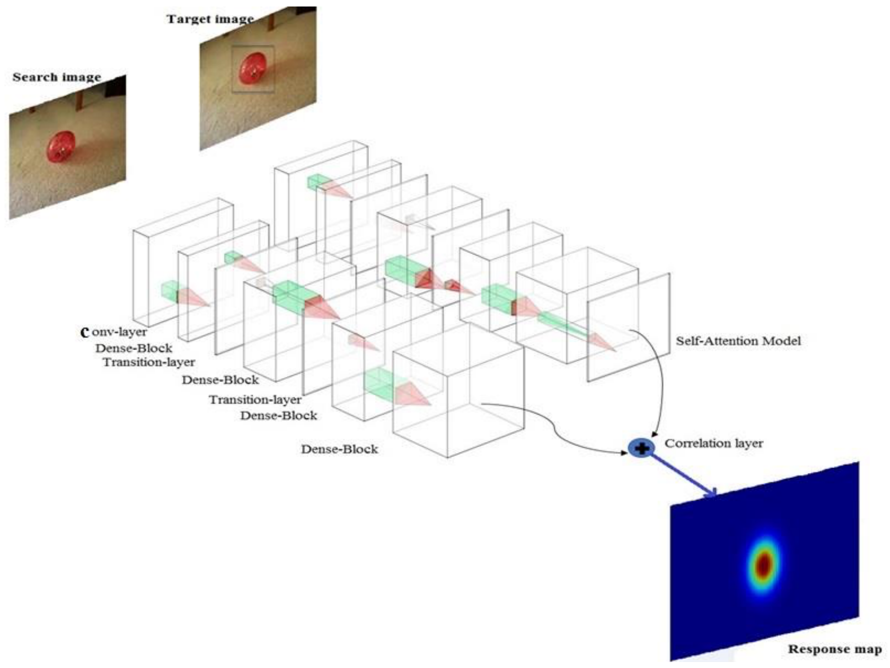


Figure 3.1: The architecture of DensSiam tracker. The target branch has the Self-Attention model to capture the semantic information during offline training.

3.4.1 Densely-Siamese Architecture

DensSiam architecture consists of two branches, the target branch and the appearance branch. The target branch architecture as follows: input-ConvNet-DenseBlock-TransitionLayer-DenseBlock-TransitionLayer-DenseBlock-DenseBlock-SelfAttention.

Dense block: Consists of Batch Normalization (BN) [20], Rectified Linear Units (ReLU), pooling and Convolution layers, all dimensions are shown in Table 3.1. Each layer in dense block takes all preceding feature maps as input and concatenate them. These connections ensure that the network will preserve the information needed from all preceding feature maps and improve the information flow between layers, and thus improves the generalization capability as shown in Fig. 3.2. In traditional Siamese the output of l^{th} layer is fed in as input to the $(l+1)^{th}$ layer we denote the output of l^{th} layer as x_l. To formulate the information flow between layers in

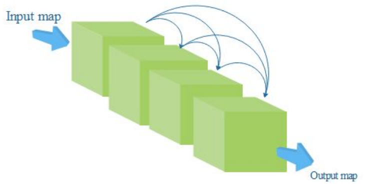


Figure 3.2: The internal structure of dense block without BN, ReLU, 1×1 conv and Pooling layers shows the connections between convolutional layers in $DenseBlock_2$

dense block lets assume that the input tensor is $x_0 \in \Re^{C\times N\times D}$. Consequently, l^{th} layer receives all feature maps according to this equation:

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]), \tag{3.1}$$

Where H_l consists of three consecutive operations, batch normalization, ReLU and a 3×3 convolution and $[x_0, x_1, ..., x_{l-1}]$ is a feature map concatenation.

Transition layer: Consists of convolutional operation, average pooling and dropout [21]. Adding dropout to dense block and transition layer decreases the risk that DensSiam overfits to negative classes. DensSiam does not use padding operation since it is a fully-convolutional and padding violates this property. Consequently, the size of feature maps in dense blocks varies and can not be concatenated. Therefore, transition layers are used to match up the size of dense blocks and concatenate them properly.

3.4.2 Self-Attention Model

Attention mechanism was used in image classification [22], multi-object tracking [23], pose estimation [24], *etc.* In addition to the new architecture of DensSiam and inspired by [25, 26] that use the attention mechanism in object detection, DensSiam architecture integrates the Self-

Attention model to target branch as shown in Fig. 3.3.

Given the input tensor $x \in \Re^{W \times H \times D}$ which is the output feature map of the target branch, Self-Attention model divides the feature map into three maps through 1×1 convolutinal operation to capture the attention, $f(x)$ and $g(x)$ can be calculated as follows:

$$f(x) = W_f \times x, \tag{3.2}$$

$$g(x) = W_g \times x, \tag{3.3}$$

Where W_f, W_g are the offline learned parameters. Attention feature map can be calculated as follows:

$$\phi = \frac{\exp(m)}{\sum_{i=1}^{N} \exp(m)}, \tag{3.4}$$

Where ϕ is the attention map weights and m= $f(x_i)^T g(x_i)$. Self-Attention feature map can be calculated as follows:

$$\sum_{i=1}^{N} (\phi \times h(x_i)), \tag{3.5}$$

Where $h(x) = W_h \times x$ and W_h is the offline learned parameter. We use logistic loss for both dense block and Self-Attention model to calculate the weights using Stochastic Gradient Descent (SGD) as follows:

$$l(y, v) = log(1 + \exp(-yv)), \tag{3.6}$$

Where v is the single score value of target-candidate pair and $y \in [-1, +1]$ its ground truth label. To calculate the loss function for the feature map we use the mean of the loss over the

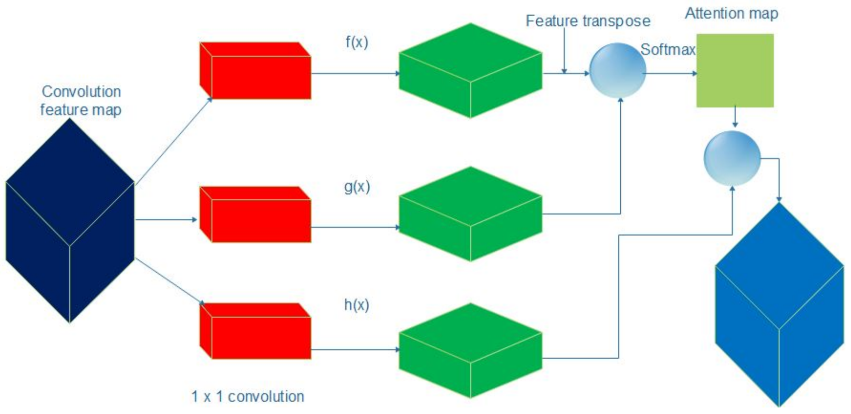


Figure 3.3: Self-Attention Model, the feature map is divided into three maps, blue circles are matrix multiplication. The input and the output tensors are the same size.

whole map as follows:

$$L(y, v) = \frac{1}{N} \sum_{n \in N} l(y[n], v[n]), \tag{3.7}$$

Finally, the search branch has the same architecture as target branch except the search branch does not have the Self-Attention model. The output of Self-Attention map and the output of the search branch are fed into correlation layer to learn the similarity function.

3.5 Experimental Results

We divided the benchmarks to two sets, the validation set which includes OTB2013, OTB2015 and the testing set which includes VOT2015 VOT2016 and VOT2017. We provide the implementation details and hyper-parameters in the next subsection.

3.5.1 Implementation Details

DensSiam is pre-trained offline from scratch on the video object detection dataset of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC15) [27]. ILSVRC15 contains 1.3 million labelled frames in 4000 sequences and it has a wide variety of objects which contribute to the generalization of the DensSiam network.

Network architecture. We adopt this architecture $input - ConvNet - DenseBlock_1 - TransitionLayer - DenseBlock_2 - TransitionLayer - DenseBlock_3 - DenseBlock_4 - SelfAttention$ as shown in Table 3.1.

Hyper-parameters settings. Training is performed over 100 epochs, each with 53,200 sampled pairs. Stochastic gradient descent (SGD) is applied with momentum of 0.9 to train the network. We adopt the mini-batches of size 8 and the learning rate is annealed geometrically at each epoch from 10^{-3} to 10^{-8}. We implement DensSiam in TensorFlow [28] 1.8 framework. The experiments are performed on a PC with a Xeon E5 2.20 GHz CPU and a Titan XP GPU. The testing speed of DensSiam is 60 fps.

Tracking settings. We adapt the scale variations by searching for the object on three scales O^s where $O = 1.0375$ and $s = \{-2, 0, 2\}$. The input target image size is 127 × 127 and the search image size is 255×255. We use the linear interpolation to update the scale with a factor 0.764.

3.5.2 Comparison with the State-of-the-Art

In this section we use VOT toolkit standard metrics [29], accuracy (A), robustness (R) and expected average overlap (EAO). Table 5.2 shows the comparison of DensSaim with MDNet [30], DeepSRDCF [13], EBT [31], BACF [32], EAST [33], Staple [34] and SiamFC [11]. The four top trackers are non-realtime trackers and we still outperform them. In terms of accuracy,

Table 3.1: Data dimensions in DensSiam.

Layers	Output size	Target branch	Search branch
$Convolution$	Tensor (8, 61, 61, 72)	7×7 conv, stride 2	7×7 conv, stride 2
$DenseBlock_1$	Tensor(8, 61, 61, 144)	$[1 \times 1 conv] \times 2$ $[3 \times 3 conv] \times 2$	$[1 \times 1 conv] \times 2$ $[3 \times 3 conv] \times 2$
$TransitionLayer$	Tensor (8, 30, 30, 36)	1×1 conv , average pool	1×1 conv , average pool
$DenseBlock_2$	Tensor(8, 30, 30, 180)	$[1 \times 1 conv] \times 4$ $[3 \times 3 conv] \times 4$	$[1 \times 1 conv] \times 4$ $[3 \times 3 conv] \times 4$
$TransitionLayer$	Tensor(8, 15, 15, 36)	1×1 conv , average pool	1×1 conv , average pool
$DenseBlock_3$	Tensor (8, 15, 15, 252)	$[1 \times 1 conv] \times 6$ $[3 \times 3 conv] \times 6$	$[1 \times 1 conv] \times 6$ $[3 \times 3 conv] \times 6$
$DenseBlock_4$	Tensor(8,9,9,128)	$[7 \times 7 conv] \times 3$	$[7 \times 7 conv] \times 3$
$Self - Attention$	Tensor(8, 9, 9, 128)	$[1 \times 1 conv] \times 3$	–

DensSiam is about 2 % higher than MDNet while it is the best second after MDNet in terms of expected average overlap. DensSiam is the highest in terms of robustness score in real-time trackers. We also report the results of our tracker on VOT2016 and VOT2017 as shown in Table 5.3 and Table 4.1. The comparison includes ECOhc [35], SiamFC [11], SiamDCF [36], Staple [34], CSRDCF++ [37]. DensSiam outperforms all trackers in terms of accuracy, robustness score and expected average overlap.

3.6 Conclusions and Future Work

This chapter proposed DensSiam, a new Siamese architecture for object tracking. DensSiam uses non-local features to represent the appearance model in such a way that allows the deep feature map to be robust to appearance changes. DensSaim allows different feature levels (e.g.

Table 3.2: Comparison with the state-of-the-art trackers including the top four non-realtime for VOT2015.

Tracker	A	R	EAO	FPS
MDNet	0.60	0.69	0.38	1
DeepSRDCF	0.56	1.05	0.32	< 1
EBT	0.47	1.02	0.31	4.4
SRDCF	0.56	1.24	0.2	5
BACF	0.59	1.56	–	35
EAST	0.57	1.03	0.34	159
Staple	0.57	1.39	0.30	80
SamFC	0.55	1.58	0.29	86
DensSiam(ours)	0.619	1.24	0.34	60

Table 3.3: Comparison with the state-of-the-art trackers on VOT2016.

Tracker	A	R	EAO	FPS
ECOhc	0.54	1.19	0.3221	60
Staple	0.54	1.42	0.2952	80
STAPLE+	0.55	1.31	0.2862	> 25
SiamRN	0.55	1.36	0.2766	> 25
GCF	0.51	1.57	0.2179	> 25
DensSiam(ours)	0.56	1.08	0.3310	60

Table 3.4: Comparison with the state-of-the-art trackers on VOT2017.

Tracker	A	R	EAO	FPS
SiamDCF	0.500	0.473	0.249	60
ECOhc	0.494	0.435	0.238	60
CSRDCF++	0.453	0.370	0.229	> 25
SiamFC	0.502	0.585	0.188	86
SAPKLTF	0.482	0.581	0.184	> 25
Staple	0.530	0.688	0.169	> 80
ASMS	0.494	0.623	0.169	> 25
DensSiam(ours)	0.540	0.350	0.250	60

low level and high level features) to flow through the network layers without vanishing gradients and improves the generalization capability . The resulting tracker greatly benefits from the Densely-Siamese architecture with Self-Attention model and substantially increases the accuracy and robustness while decreasing the number of shared network parameters. The architecture of DensSiam can be extended to other tasks of computer vision such as object verification, recognition and detection since it is general Siamese framework.

References

[1] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[2] A. Kendall, M. Grimes, and R. Cipolla, “Posenet: A convolutional network for real-time 6-dof camera relocalization,” in *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2938–2946.

[3] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, “Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4207–4215.

[4] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, “Deepdriving: Learning affordance for direct perception in autonomous driving,” in *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2722–2730.

[5] Y. Wu, J. Lim, and M.-H. Yang, “Object tracking benchmark,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[6] K. Alahari, A. Berg, G. Hager, J. Ahlberg, M. Kristan, J. Matas, A. Leonardis, L. Cehovin, G. Fernandez, T. Vojir *et al.*, “The thermal infrared visual object tracking vot-tir2015 challenge results,” in *Computer Vision Workshop (ICCVW), 2015 IEEE International Conference on*. IEEE, 2015, pp. 639–651.

[7] M. Mueller, N. Smith, and B. Ghanem, “Context-aware correlation filter tracking,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, 2017, pp. 1396–1404.

[8] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, “Learning spatially regularized correlation filters for visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4310–4318.

[9] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[10] Y. Li and J. Zhu, “A scale adaptive kernel correlation filter tracker with feature integration,” in *European Conference on Computer Vision*. Springer, 2014, pp. 254–265.

[11] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, “Fully-convolutional siamese networks for object tracking,” in *European conference on computer vision*. Springer, 2016, pp. 850–865.

[12] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, “End-to-end representation learning for correlation filter based tracking,” in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5000–5008.

[13] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, “Convolutional features for correlation filter based visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 58–66.

[14] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, “Hierarchical convolutional features for visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3074–3082.

[15] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, “A survey of appearance models in visual object tracking,” *ACM transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, p. 58, 2013.

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

[17] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, “Learning dynamic siamese network for visual object tracking,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1–9.

[18] R. Tao, E. Gavves, and A. W. Smeulders, “Siamese instance search for tracking,” in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 1420–1429.

[19] A. He, C. Luo, X. Tian, and W. Zeng, “A twofold siamese network for real-time object tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4834–4843.

[20] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” in *ICML*, vol. 37, 2015, pp. 448–456.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, “Residual attention network for image classification,” *arXiv preprint arXiv:1704.06904*, 2017.

[23] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, “Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism,” in *2017 IEEE International Conference on Computer Vision (ICCV).(Oct 2017)*, 2017, pp. 4846–4855.

[24] W. Du, Y. Wang, and Y. Qiao, “Rpan: An end-to-end recurrent pose-attention network for action recognition in videos,” in *IEEE International Conference on Computer Vision*, vol. 2, no. 4, 2017.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010.

[26] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” *CVPR*, 2018.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[28] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin *et al.*, “Tensorflow: large-scale machine learning on heterogeneous distributed systems. arxiv preprint (2016),” *arXiv preprint arXiv:1603.04467*.

[29] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, “The visual object tracking vot2016 challenge results,” in *European Conference on Computer Vision Workshop*, 2016.

[30] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[31] G. Zhu, F. Porikli, and H. Li, “Beyond local search: Tracking objects everywhere with

instance-specific proposals," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 943–951.

[32] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, 2017, pp. 21–26.

[33] C. Huang, S. Lucey, and D. Ramanan, "Learning policies for adaptive tracking with deep feature cascades," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 105–114.

[34] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1401–1409.

[35] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, 2017, pp. 21–26.

[36] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu, "Dcfnet: Discriminant correlation filters network for visual tracking," *arXiv preprint arXiv:1704.04057*, 2017.

[37] A. Lukezic, T. Vojir, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017.

Chapter 4

DomainSiam

4.1 Abstract

Visual object tracking is a fundamental task in the field of computer vision. Recently, Siamese trackers have achieved *state-of-the-art* performance on recent benchmarks. However, Siamese trackers do not fully utilize semantic and objectness information from pre-trained networks that have been trained on image classification task. Furthermore, the pre-trained Siamese architecture is sparsely activated by the category label, which leads to unnecessary calculations and overfitting. In this chapter, we propose to learn a Domain-Aware that fully utilizes semantic and objectness information while producing a class-agnostic features using a ridge regression network. Moreover, to reduce the sparsity problem, we solve the ridge regression problem with a differentiable weighted-dynamic loss function. Our tracker, dubbed *DomainSiam*, improves the feature learning in the training phase and generalization capability to other domains. Extensive experiments are performed on five tracking benchmarks, including OTB2013 and OTB2015, for a validation set as well as VOT2017, VOT2018, LaSOT, TrackingNet, and GOT10k for a

testing set. *DomainSiam* achieves a *state-of-the-art* performance on these benchmarks while running at 53 FPS.

4.2 Introduction

Tracking is a fundamental task in many computer vision tasks such as surveillance [1], computer interactions [2] and image understanding [3]. The objective of tracking is to find the trajectory of the object of interest over time. This is a challenge since the object of interest undergoes appearance changes such as occlusions, motion blur, and background cluttering [4, 5]. Recent deep trackers such as CFNet [6] and DeepSRDCF [7] use pre-trained networks that have been trained on image classification or object recognition.

In recent years, convolutional neural networks (CNNs) have achieved superior performance against hand-crafted trackers (e.g., CACF [8], SRDCF [9], KCF [10] and SAMF [11]). Siamese trackers such as SiamFC [12], CFNet [6], SiamRPN [13], and DensSiam [14] learn a similarity function to separate the foreground from its background. However, Siamese trackers do not fully utilize semantic and objectness information from pre-trained networks that have been trained on image classification. In image classification, the class categories of the objects are pre-defined, while in object tracking tasks, the tracker needs to be class-agnostic while benefiting from semantic and objectness information. Moreover, the image classification increases the inter-class differences while forcing the features to be insensitive to intra-class changes [15].

In this chapter, we propose DomainSiam to learn Domain-Aware, which fully utilizes semantic and objectness information from a pre-trained network. DomainSiam consists of DensSiam with a self-attention module [14] as a backbone network and a regression network to select the most discriminative convolutional filters to leverage the semantic and objectness information.

Moreover, we develop a differentiable weighted-dynamic domain loss function to train the regression network. The developed loss function is monotonic, dynamic, and smooth with respect to its hyper-parameters, which can be reduced to l_1 *or* l_2 during the training phase. On the other hand, the shrinkage loss function [16] is static, and it can not be adapted during the training phase. Most regression networks solve the regression problem with static loss such as the closed-form solution if the input to the network is not high-dimensional or minimizing l_2. The results will be made available [1].

To summarize, the main contributions of this chapter are three-fold.

- A novel architecture is proposed for object tracking to capture the Domain-Aware features with semantic and objectness information. The proposed architecture enables the features to be robust to appearance changes. Moreover, it decreases the sparsity problem, as it produces the most important feature space. Consequently, it decreases the overhead calculations.

- A differentiable weighted-dynamic domain loss function is developed specifically for visual object tracking to train the regression network to extract the domain channels that are activated by target category. The developed loss is monotonic with respect to its hyper-parameters, and this will be useful in case of high-dimensional data and non-convexity. Consequently, this will increase the performance of the tracker.

- The proposed architecture tackles the generalization capability from one domain to another domain (e.g., from ImageNet to VOT datasets).

[1] https://vip-mun.github.io/DomainSiam

The rest of the chapter is organized as follows. Related work is presented in section 4.3. Section 4.4 details the proposed approach. We present the experimental results in Section 4.5. Finally, section 4.6 concludes the chapter.

4.3 Related work

In this section, we firstly introduce the *state-of-the-art* Siamese-based trackers. Then, we briefly introduce the gradient-based localization guidance.

Siamese-based Trackers

Recently, Siamese-based trackers have received significant attention, especially in realtime tracking due to their balanced accuracy and speed. However, Siamese-based trackers drift to the background due to lack of semantic and objectness information about the positive samples. Siamese-based trackers always provide a heat map which encodes the most important channels for the object category as well as the sparse channels. Consequently, the heat map is sparsely activated by the category label. In general, a Siamese network consists of two branches: the target branch and the search branch, and both branches share the same parameters. The score map, which indicates the position of the object of interest, is generated by the last cross-correlation layer.

The first Siamese network was first proposed in [17] for signature verification. The pioneering work that uses Siamese in object tracking is SiamFC [12]. SiamFC searches the target image in the search image. Siamese Instance Search [18] proposed SINT, which has a query branch and a search branch, and the backbone of this architecture is inherited from AlexNet

[19]. CFNet [6] enabled SiamFC to be re-trained once per frame instead of using an offline pre-trained network. CFNet integrated a correlation layer to back-propagate gradients through an online learning. Moreover, CFNet improved SiamFC by adding a correlation layer to the target branch. SA-Siam [20] proposed two Siamese networks: the first network encodes the semantic information, and the second network encodes the appearance model. This is different from our architecture, which has only one Siamese network. SiamRPN [21] formulated the tracking problem as a local one-shot detection. SiamRPN consists of a Siamese network as a feature extractor and a region proposal network that includes the classification branch and regression branch. DensSiam [14] used the Densely-Siamese architecture to make the Siamese network deeper while maintaining the performance of the network. DensSiam allows the low-level and high-level features to flow within layers without the vanishing gradients problems. Moreover, a self-attention mechanism was integrated to force the network to capture the non-local features. SiamMask [22] improved the offline training of Siamese networks by using augmentation loss to produce a binary segmentation mask. In addition, the binary segmentation mask locates the object of interest accurately. ATOM [23] formulated the tracking as a target estimation problem. ATOM proposed a Siamese network with explicit components for target estimation and classification. The components are trained offline to maximize the overlapping between the estimated bounding box and the target. Siamese-based trackers provide a heat map which is sparsely activated by the category label due to lack of the semantic and objectness information.

Gradient-based Localization Guidance

It turns out that the gradient can be used to determine the importance of each channel in the heat map in Siamese networks. Moreover, the global average pooling of the gradients acts as Attention to locate the target from the heat map. In this category of learning, the objective is to determine the most important channel of the network with respect to the object category. In an object classification task, each category activates a set of certain channels. Grad-CAM [24] used a gradient of any target logit (e.g., ”cat”) and, using this gradient, determined the active category channel for this logit. The work in [25] demonstrated that the global average pooling of the gradients is implicitly acting as Attention for the network; consequently, it can locate the object of interest accurately.

4.4 Proposed Approach

We propose DomainSiam for visual object tracking. The complete pipeline is shown in Fig. 4.1. The DensSiam with the Self-Attention network is used as a feature extractor; however, in any Siamese network, these features do not fully utilize the semantic and objectness information. Furthermore, the channels in Siamese networks are sparsely activated. We use the ridge regression network with a differentiable weighted-dynamic loss function to overcome the previous problems.

4.4.1 Ridge Regression Network with Domain-Aware Features

In Fig. 4.1, the pipeline is divided into three blocks: the input block to the target branch and the search branch; the DensSiam block, which has the same architecture in [14]; and the ridge regression network. The DensSiam network produces two feature maps for target and search images, respectively. Imbalanced distribution of the training data makes the feature maps produced by Siamese networks less discriminative, as there is a high number of easy samples com-

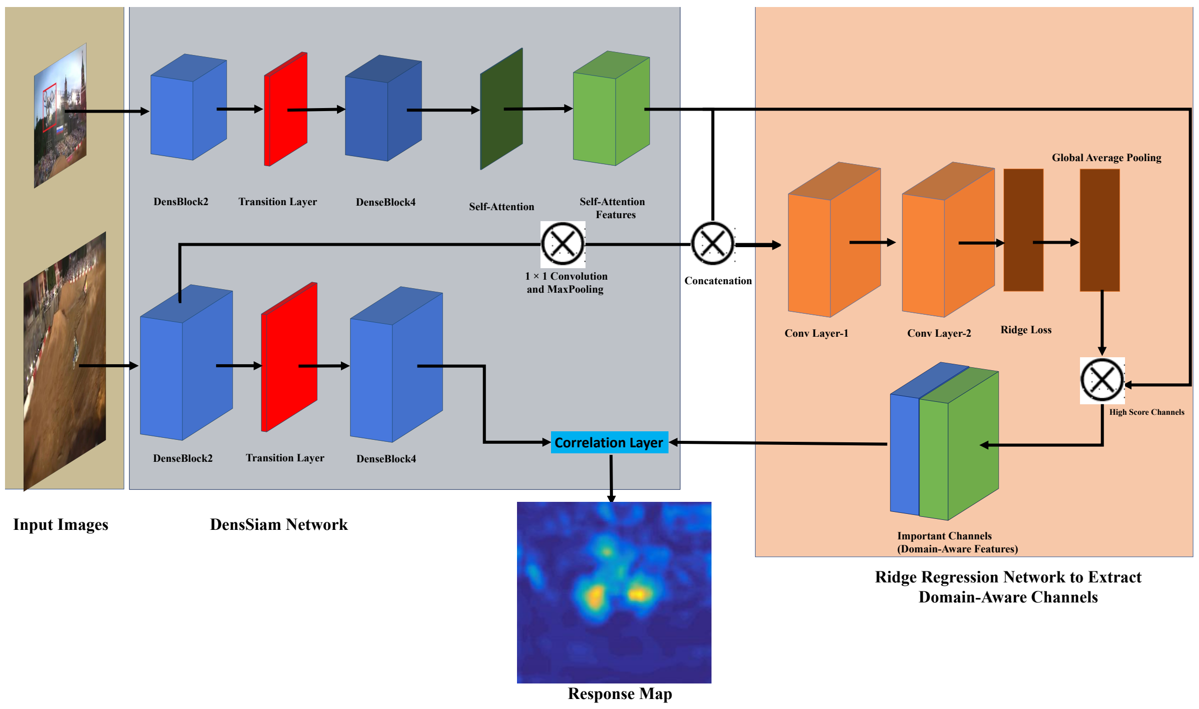


Figure 4.1: The architecture of DomainSiam tracker. It consists of three blocks: the input images block, which includes the target image and search image; the DensSiam network with a Self-Attention module at the end of the target branch; and the Ridge Regression Network, which highlights the important channels and produces the Domain-Aware features. The response map is produced by the correlation layer, which is the final layer. The correlation layer calculates the correlation between the Domain-Aware channels and search branch features and is represented by DenseBlock4.

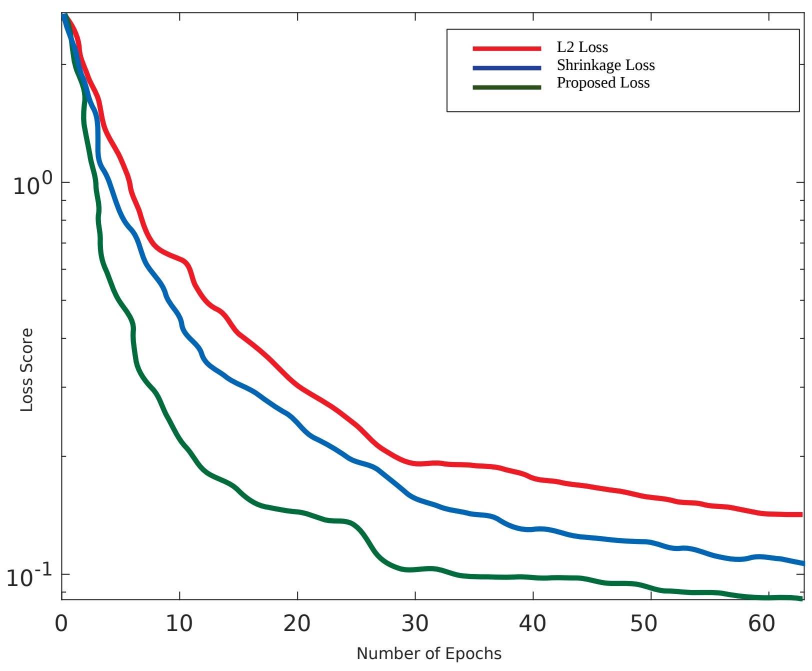


Figure 4.2: A comparison of convergence speed on L_2 loss, Shrinkage loss [16], and our proposed loss function. The average loss is calculated on a batch of eight samples on VOT2018 [26] dataset.

pared to the hard samples. Siamese networks use pre-trained networks that have been trained on other tasks (e.g., classification and recognition). These networks increase inter-class differences and are also insensitive to intra-class variations. Consequently, this property decreases the performance of Siamese networks, as the tracker needs to be more discriminative to the same object category. Moreover, the pre-trained network is sparsely activated by the object category. In other words, in the feature channels/maps there are only a few active channels that correspond to the object category. The regression network in Fig. 4.1 highlights the importance of each channel in the feature map to the object of interest and discards the others.
In Fig. 4.1, the ridge regression network regresses all samples in the input image patch to their soft labels by optimizing the following objective function.

$$\underset{w}{arg\,min} \|W * X_{i,j} - Y(i,j)\|^2 + \lambda \|W\|^2 \tag{4.1}$$

Where $*$ denotes the convolution operation, W is the weight of the regression network, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the input features and $\mathbf{Y} \in \mathbb{R}^{N \times D}$ is the soft label. Gaussian distribution is used as a soft label map, and its centre is aligned to the target center and $\lambda > 0$ is the regularization parameter.

$$Y(i,j) = e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{4.2}$$

Where (i,j) is the location corresponding to the target location and σ is the Gaussian kernel width. The closed-form analytic solution for equation 4.1 is defined as

$$W = \left(X^\top X + \lambda I\right)^{-1} X^\top Y \tag{4.3}$$

The optimal solution of W can be achieved by equation 4.3; however, solving this equation is computationally expensive as $X^\top X \in \mathbb{R}^{D \times D}$. Instead, we use the ridge regression network with the proposed loss function to solve equation 4.1.

4.4.2 Ridge Regression Optimization

In Fig. 4.1, the ridge regression network consists of two convolutional layers, ridge loss and the global average pooling. The global average pooling encourages the proposed loss function to localize the object of interest accurately compared to the global max pooling. It is worth mentioning that both global average pooling and global max pooling have similar performances on object classification tasks. As shown in Fig. 4.1, in the last block, the Domain-Aware feature space is calculated by

$$\delta_i = GAP(\partial L/\partial F_i) \tag{4.4}$$

Where δ is the Domain-Aware non-sparse features; GAP is the global average pooling; L is the domain-dynamic loss function, which will be discussed later; and F is the input feature channel of the i^{th} channel to the ridge regression network. Let the objective function of the ridge regression network be x

$$x = \|W * X_{i,j} - Y(i,j)\|^2 + \lambda\|W\|^2 \tag{4.5}$$

We propose a differentiable weighted-dynamic loss function for visual object tracking to solve equation 4.5. This is inspired by [27], who uses a general loss function for the variational autoencoder, monocular depth estimation, and global registration, as follows.

$$L(x,\alpha) = \frac{|\alpha-2|}{\alpha} e^{ay} \left(\left(\frac{x^2}{|\alpha-2|} + 1 \right)^{\alpha/2} - 1 \right) \tag{4.6}$$

where $a \in [0,1]$ is a hyper-parameter, y is the regression target of a sample, and $\alpha \in \mathbb{R}$ is the parameter that controls the robustness of the loss. The exponent term in this loss function tackles the imbalanced distribution of the training set by assigning a higher weight to hard samples. The imbalanced data occurs when the number of easy samples (background) is extremely higher than the hard samples (foreground).

The advantage of this loss function over equation 4.1 and equation 4.3 is that it can automatically adjust the robustness during the training phase. This advantage comes from the α parameter. For example, at $\alpha = 2$ the equation 4.6 becomes L_2

$$\lim_{\alpha \to 2} L(x, \alpha) = \frac{e^{ay}}{2} x^2 \tag{4.7}$$

Similarly, when $\alpha = 1$, the equation 4.6 becomes L_1

$$L(x, \alpha) = (\sqrt{x^2 + 1}) e^{ay} - 1 \tag{4.8}$$

Another advantage of equation 4.6 is becoming Lorentzian loss function [28] by allowing $\alpha = 0$ as follows

$$\lim_{x \to 0} L(x, \alpha) = \log\left(\frac{1}{2} x^2 + 1\right) e^{ay} \tag{4.9}$$

As noted before, the proposed loss function is dynamic, which allows the network to also learn a robust representation. The gradient of the equation 4.6 with respect to α is always positive. Consequently, this property makes the loss monotonic with respect to α and useful for non-convex optimization.

$$\frac{\partial L}{\partial \alpha}(x, \alpha) \geq 0 \tag{4.10}$$

The final proposed loss function is given by

$$L(x, \alpha) = \begin{cases} \frac{e^{ay}}{2} x^2 & \text{if } \alpha = 2 \\ \log\left(\frac{1}{2}(x)^2 + 1\right) e^{ay} & \text{if } \alpha = 0 \\ (1 - \exp\left(-\frac{1}{2}(x)^2\right)) e^{ay} & \text{if } \alpha = -\infty \\ \frac{|\alpha - 2|}{\alpha} e^{ay} \left(\left(\frac{(x)^2}{|\alpha - 2|} + 1\right)^{\alpha/2} - 1\right) & \text{otherwise} \end{cases} \tag{4.11}$$

Fig. 4.2 shows that the optimization over the proposed loss function achieves faster convergence speed, while in the Shrinkage loss function proposed in [16] and the original ridge regression loss function 4.1 (l_2), the convergence speed is slower. The importance of each channel in the feature map is calculated by plugging equation 4.11 into equation 4.4. It is worth noting that the output feature map of the ridge regression network contains only the activated channels that have the most semantic and objectness information corresponding to the object category. The Domain-Aware features and the feature channels from denseBlock4 are fed into the correlation layer to calculate the similarity and produce the response map.

4.5 Experimental Results

The benchmarks are divided into two categories: the validation set including OTB2013 [29] and OTB2015 [4] and the testing set including VOT2017 [30], VOT2018 [26], and GOT10k [31]. We introduce the implementation details in the next sub-section and then we compare the proposed tracker to the *state-of-the-art* trackers.

4.5.1 Implementation Details

We used the pre-trained DensSiam network (DenseBlock2 and DenseBlock4) that has been trained on Large Scale Visual Recognition Challenge (ILSVRC15) [32]. ILSVRC15 has over 4000 sequences with approximately 1.3 million frames and their labels . DomainSiam, which has been trained on 1000 classes, can benefit from this class diversity. We implemented DomainSiam in Python using a PyTorch framework [33]. Experiments are performed on Linux with a Xeon E5 @2.20 GHz CPU and a Titan XP GPU. The testing speed of DomainSiam is 53 FPS, which is beyond realtime speed.

Table 4.1: Comparison with the state-of-the-art trackers on VOT2017 in terms of Accuracy (A), expected Average Overlap (EAO), and Robustness (R).

Tracker	A↑	EAO↑	R ↓	FPS
CSRDCF++	0.453	0.229	0.370	> 25
SAPKLTF	0.482	0.184	0.581	> 25
Staple	0.530	0.169	0.688	> 80
ASMS	0.494	0.169	0.623	> 25
SiamFC	0.502	0.188	0.585	86
SiamDCF	0.500	0.473	0.249	60
ECOhc	0.494	0.435	0.238	60
DensSiam	0.540	0.350	0.250	60
DomainSiam(proposed)	**0.562**	**0.374**	**0.201**	53

Training. The ridge regression network is trained with its proposed loss function separately from the Siamese network with 70 epochs. The highest scores associated with 100 channels are selected as the Domain-Aware features. The training is applied with a momentum of 0.9, a batch size of 8 images, and the learning rate is annealed geometrically at each epoch from 10^{-3} to 10^{-8}.

Tracking Settings. The initial scale variation is O^s where $O = 1.0375$ and $s = \{-2, 0, 2\}$. We adopt the target image size of 127×127 and the search image size of 255×255 with a linear interpolation to update the scale with a factor of 0.435.

Table 4.2: Comparison with *state-of-the-art* trackers on VOT2018 in terms of Accuracy (A), expected Average Overlap (EAO), and Robustness (R).

Tracker	A↑	EAO↑	R↓	FPS
ASMS [34]	0.494	0.169	0.623	25
SiamRPN [13]	0.586	0.383	0.276	160
SA_Siam_R [20]	0.566	0.337	0.258	50
FSAN [26]	0.554	0.256	0.356	30
CSRDCF [35]	0.491	0.256	0.356	13
SiamFC [12]	0.503	0.188	0.585	86
SAPKLTF [26]	0.488	0.171	0.613	25
DSiam [36]	0.215	0.196	0.646	25
ECO [37]	0.484	0.280	0.276	60
DomainSiam(proposed)	**0.593**	**0.396**	**0.221**	53

4.5.2 Comparison with State-of-the-Art Trackers

In this section, we use five benchmarks to evaluate DomainSiam against *state-of-the-art* trackers. We use VOT2017 [30], VOT2018 [26], LaSOT [46], TrackingNet [47], and GOT10k [31].

Results on VOT2017 and VOT2018

We used the standard metrics on short-term challenge on the VOT dataset. The results on the VOT dataset shown in Table 4.1 and Table 5.4 are given by the VOT-Toolkit. DomainSiam outperforms the *state-of-the-art* trackers listed in both tables. It is worth mentioning that DomainSiam is about 2% higher than the DensSiam tracker in terms of Accuracy (A) and Expected

Table 4.3: Comparisons with *state-of-the-art* trackers on TrackingNet dataset in terms of the Precision (PRE), Normalized Precision (NPRE), and Success.

Tracker	PRE ↑	NPRE ↑	SUC.↑
Staple_CA [8]	0.468	0.605	0.529
BACF [38]	0.461	0.580	0.523
MDNet [39]	0.565	0.705	0.606
CFNet [6]	0.533	0.654	0.578
SiamFC [12]	0.533	0.663	0.571
SAMF [11]	0.477	0.598	0.504
ECO-HC [37]	0.476	0.608	0.541
Staple [40]	0.470	0.603	0.528
ECO [37]	0.492	0.618	0.554
CSRDCF [35]	0.480	0.622	0.534
DomainSiam(proposed)	**0.585**	**0.712**	**0.635**

Average Overlap (EAO) in Table 4.1 while running at 53 frames per second. Table 5.4 shows that DomainSiam is ranked as the best tracker in terms of accuracy with 0.593 and gain of 0.7 % compared to the second-best tracker, which is SiamRPN. In terms of expected average overlap, DomainSiam is ranked as the best tracker with score of 0.396 while the score of the second-best tracker is 0.383 with gain of 1.3 %. In terms of robustness, the number of failures per sequence, DomainSiam achieves the best score of 0.221, which is about 5% higher than the second-best tracker (SiamRPN) while working in realtime speed.

Results on TrackingNet Dataset

Table 4.4: Comparison with *state-of-the-art* trackers on LaSOt dataset in terms of the Normalized Precision and Success.

Tracker	Norm. Prec. (%)↑	Success (%)↑
MDNet [39]	46.0	39.7
DaSiam [41]	49.6	41.5
STRCF [42]	34.0	30.8
SINT [18]	35.4	31.4
StrucSiam [43]	41.8	33.5
SiamFC [12]	42.0	33.6
VITAL [44]	45.3	39.0
ECO [45]	33.8	32.4
DSiam [36]	40.5	33.3
DomainSiam(proposed)	53.7	43.6

Table 4.5: Comparison *state-of-the-art* trackers on GOT10k dataset in terms of Average Overlap (AO), and Success Rates (SR) at overlap thresholds of 0.50 and 0.75.

TRACKER	DomainSiam (proposed)	CFNet	SiamFC	GOTURN	CCOT	ECO	HCF	MDNet
AO	**0.414**	0.374	0.348	0.347	0.325	0.316	0.315	0.299
SR(0.50)	**0.451**	0.404	0.353	0.375	0.328	0.309	0.297	0.303
SR(0.75)	**0.214**	0.144	0.098	0.124	0.107	0.111	0.088	0.099

This is a large-scale dataset that was collected from YouTube videos. Table 7.6 shows that DomainSiam outperforms MDNet, which is the second-best tracker on the TrackingNet dataset with 2% in terms of precision and about 3% in terms of success. DomainSiam outperforms all other trackers on the TrackingNet dataset.

Results on the LaSOT Dataset

The average sequence length in this dataset is about 2500 frames. Table 7.8 shows that DomainSiam achieves the best success score with over 2% from the second-best tracker (DaSiam). Our tracker significantly outperforms DaSiam with 4% in terms of normalized precision.

Results on GOT10k Dataset

This dataset has 180 test sequences. We tested the proposed tracker against seven trackers as shown in Table 7.7. DomainSiam outperforms CFNet, which is the best tracker in terms of Average Overlap (AO) with 4%. It is worth noting that DomainSiam achieves the best performance among all trackers in terms of Success Rate (SR) at thresholds of 0.50 and 0.75.

4.6 Conclusions and Future Work

In this chapter, we introduced DomainSiam tracker, a Siamese with a ridge regression network that fully utilizes semantic and objectness information for visual object tracking while also producing a class-agnostic features. We developed a differentiable weighted-dynamic loss function to solve the ridge regression problem. The developed loss function improves the feature learning, as it automatically adjusts the robustness during the training phase. Furthermore, it utilizes the activated channels that correspond to the object category label. The proposed architecture decreases the sparsity problem in Siamese networks and provides an efficient Domain-Aware feature space that is robust to appearance changes. DomainSiam does not need to be re-trained from scratch, as the ridge regression network with the proposed loss function is trained separately from the Siamese network. DomainSiam with the proposed loss function exhibits a superior convergence speed compared to other loss functions. The ridge regression network with the proposed loss function can be extended to other tasks such as object detection and semantic segmentation.

References

[1] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2938–2946.

[2] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4207–4215.

[3] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

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

[5] K. Alahari, A. Berg, G. Hager, J. Ahlberg, M. Kristan, J. Matas, A. Leonardis, L. Cehovin, G. Fernandez, T. Vojir *et al.*, "The thermal infrared visual object tracking vot-tir2015 challenge results," in *Computer Vision Workshop (ICCVW), 2015 IEEE International Conference on*. IEEE, 2015, pp. 639–651.

[6] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, “End-to-end representation learning for correlation filter based tracking,” in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5000–5008.

[7] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, “Convolutional features for correlation filter based visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 58–66.

[8] M. Mueller, N. Smith, and B. Ghanem, “Context-aware correlation filter tracking,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, 2017, pp. 1396–1404.

[9] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, “Learning spatially regularized correlation filters for visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4310–4318.

[10] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[11] Y. Li and J. Zhu, “A scale adaptive kernel correlation filter tracker with feature integration,” in *European Conference on Computer Vision*. Springer, 2014, pp. 254–265.

[12] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, “Fully-convolutional siamese networks for object tracking,” in *European conference on computer vision*. Springer, 2016, pp. 850–865.

[13] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, “High performance visual tracking with siamese

region proposal network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8971–8980.

[14] M. M. Mohamed, "Denssiam: End-to-end densely-siamese network with self-attention model for object tracking," in *Advances in Visual Computing: 13th International Symposium, ISVC 2018, Las Vegas, NV, USA, November 19–21, 2018, Proceedings*, vol. 11241. Springer, 2018, p. 463.

[15] X. Li, C. Ma, B. Wu, Z. He, and M.-H. Yang, "Target-aware deep tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1369–1378.

[16] X. Lu, C. Ma, B. Ni, X. Yang, I. Reid, and M.-H. Yang, "Deep regression tracking with shrinkage loss," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 353–369.

[17] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[18] R. Tao, E. Gavves, and A. W. Smeulders, "Siamese instance search for tracking," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 1420–1429.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[20] A. He, C. Luo, X. Tian, and W. Zeng, “A twofold siamese network for real-time object tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4834–4843.

[21] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, “High performance visual tracking with siamese region proposal network,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[22] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, “Fast online object tracking and segmentation: A unifying approach,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1328–1338.

[23] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, “Atom: Accurate tracking by overlap maximization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4660–4669.

[24] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 618–626.

[25] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[26] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey *et al.*, “The sixth visual object tracking vot2018 challenge results,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[27] J. T. Barron, “A general and adaptive robust loss function,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4331–4339.

[28] M. J. Black and P. Anandan, “The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields,” *Computer vision and image understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[29] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[30] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Hager, A. Lukezic, A. Eldesokey *et al.*, “The visual object tracking vot2017 challenge results,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1949–1972.

[31] L. Huang, X. Zhao, and K. Huang, “Got-10k: A large high-diversity benchmark for generic object tracking in the wild,” *arXiv preprint arXiv:1810.11981*, 2018.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[33] A. Paszke, S. Gross, S. Chintala, and G. Chanan, “Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration,” *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, vol. 6, 2017.

[34] T. Vojir, J. Noskova, and J. Matas, “Robust scale-adaptive mean-shift for tracking,” *Pattern Recognition Letters*, vol. 49, pp. 250–258, 2014.

[35] A. Lukezic, T. Vojir, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017.

[36] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1–9.

[37] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, 2017, pp. 21–26.

[38] H. K. Galoogahi, A. Fagg, and S. Lucey, "Learning background-aware correlation filters for visual tracking," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA*, 2017, pp. 21–26.

[39] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[40] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, "Staple: Complementary learners for real-time tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1401–1409.

[41] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 101–117.

[42] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang, "Learning spatial-temporal regular-

ized correlation filters for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4904–4913.

[43] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu, "Structured siamese network for real-time visual tracking," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 351–366.

[44] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang, "Vital: Visual tracking via adversarial learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8990–8999.

[45] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *CVPR*, 2017.

[46] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, "Lasot: A high-quality benchmark for large-scale single object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5374–5383.

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

Chapter 5

DP-Siam: Dynamic Policy Siamese Network for Robust Object Tracking

5.1 Abstract

Balancing the trade-off between real-time performance and accuracy in object tracking is a major challenge. In this chapter, a novel dynamic policy gradient Agent-Environment architecture with Siamese network (DP-Siam) is proposed to train the tracker to increase the accuracy and the expected average overlap while performing in real-time. DP-Siam is trained offline with reinforcement learning to produce a continuous action that predicts the optimal object location. DP-Siam has a novel architecture that consists of three networks: an Agent network to predict the optimal state (bounding box) of the object being tracked, an Environment network to get the Q-value during the offline training phase to minimize the error of the loss function, and a Siamese network to produce a heat-map. During online tracking, the Environment network acts as a verifier to the Agent network action. Extensive experiments are performed on six

widely used benchmarks: OTB2013, OTB50, OTB100, VOT2015, VOT2016 and VOT2018. The results show that DP-Siam significantly outperforms the current state-of-the-art trackers.

5.2 Introduction

Visual object tracking is a fundamental task in many computer vision topics such as image understanding [1], surveillance [2], human-computer interaction [3], and autonomous driving [4]. Recently, Siamese networks have received more attention in visual tracking because of their outstanding performance (e.g., [5–9]); however, they have a continuous trade-off between real-time performance and accuracy. Many trackers focus on increasing accuracy at the expense of real-time performance. For example, many trackers increase their accuracy by adopting a search strategy such as iterative search [10], tree search [11, 12], Bayesian processes [13], or bandit-based schemes [14]. However, these trackers are far from real-time performance. On the other hand, [5, 7, 15] perform in real time (100 fps), but their accuracy is low (A≈0.24).

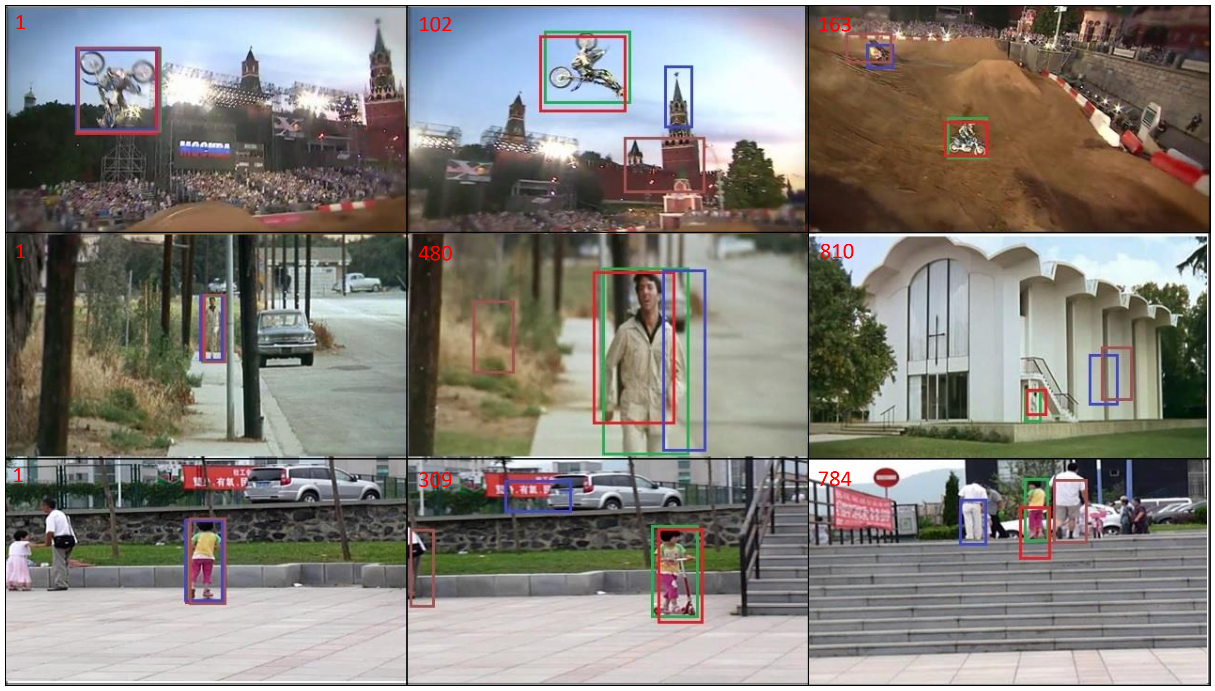
In recent years, Siamese networks have been widely used in visual object tracking; for example, in VOT2018, four out of the top nine trackers are Siamese-based. The Siamese network learns the similarity function between the template image (object of interest) and the search image (entire image), which is a very strong feature in Siamese-based trackers. We participated in the VOT2018 challenge with our previous work, DensSiam [16], and achieved good results. However, we noticed that the Expected Average Overlap (EAO), which measures the overlap between ground truth and the predicted bounding box, is relatively low in all trackers, especially in situations in which the appearance model significantly changes. Moreover, we noticed that the Robustness (R), which measures how many times the tracker fails and needs to be restarted in a sequence, is also low. Our motivation in this chapter is to address these two issues by adding an Agent-Environment framework to the Siamese network. The Agent-Environment framework uses an experience table and forces dynamic hyper-parameters selection to increase the EAO and lower the number of failures (R). To validate our motivation, the difference between our previous Siamese tracker (DensSiam) and the novel proposed architecture (DP-Siam) is shown in Table 5.6. The results of the proposed tracker (DP-Siam) are compared against current state-of-the-art trackers in Tables 5.2-5.4 on the VOT datasets, Table 5.8 on the OTB benchmark, and Table 5.10 on the UAV datasets.

In this chapter, we present a novel Siamese Agent-Environment architecture to achieve high accuracy (A) and Expected Average Overlap (EAO), by choosing the optimal state (bounding box) of the current object being tracked while performing in real time as shown in Fig. 5.1. The Agent-Environment is trained end-to-end using reinforcement learning, while Siamese learns the similarity between the search image and the template patch.

To summarize, the main contribution of this work is:

- Firstly, we propose DP-Siam, a novel dynamic Siamese Agent-Environment architecture that formulates the tracking problem with reinforcement learning. DP-Siam produces a continuous action that predicts the optimal object location. DP-Siam has a novel architecture that consists of three networks: an Agent network to predict the optimal state of the object being tracked, an Environment network to get the Q-value during the offline training phase to minimize the error of the loss function, and a Siamese network to produce a heat-map. The Environment network acts as a verifier to the action of the Agent network during online tracking.

- Secondly, the proposed architecture allows the tracker to dynamically select the hyper-parameters in each frame instead of the traditional method of fixing their values for the



– Ground truth – DP-Siam (proposed) – SiamFC – KCF

Figure 5.1: Sample results of the proposed DP-Siam tracker compared to Siam-Fc and KCF trackers. The rows show 3 different sequences, while the columns show the trackers output over time (the frame number is printed on the top left corner of the image).

entire dataset, which to the best knowledge of the authors, has not been done before.

- Finally, the design of the proposed architecture increases the generalization to other domains (e.g. from ImageNet to VOT datasets).

The results and the pre-trained network are publicly available. [1] The rest of the chapter is organized as follows. We first introduce related work in Section 5.3. Section 5.4 details the proposed approach. We present the experimental results in Section 5.5. Finally, Section 5.6 concludes the chapter and presents future directions.

5.3 Related work

Recently, deep learning has been widely used in visual object tracking. Siamese-based trackers formulate the tracking problem as a similarity learning function [6]. Reinforcement learning-based trackers formulate the tracking problem as action-decision [17–20]. Siamese trackers consist of two branches that have the same shared parameters. The basic branch extracts the template, while the other finds the candidate patches in the search image.

Siamese-based Tracking:

A Siamese network was first proposed in [21]. A Siamese network consists of two branches with shared parameters: the target branch and the search branch, as shown in the bottom left part of Fig. 5.3 (Siamese network). The last layer of the Siamese network is the cross-correlation layer. The cross-correlation layer generates the score map, which indicates the location of the object of interest. The training is done by positive and negative pairs using the logistic loss objective function.

$$\ell(y, v) = \log(1 + \exp(-yv)) \tag{5.1}$$

[1] https://vip-mun.github.io/DpSiam/

where v is the real-valued score of a single target candidate pair and y $\in \{+1, -1\}$ is its ground-truth label. This will produce a map of scores $v : \mathcal{D} \to \mathbb{R}$, which effectively locates the center of the bounding box. Object tracking can be modeled as similarity learning. The Siamese network calculates the similarity between the target input patch and a much larger search image at all translated sub-windows. Consequently, a score map will be generated, and the object of interest can be located at the highest similarity score [6].

A pioneering work of using Siamese networks in tracking is presented in SiamFC [6], which searches for the patch most similar to the template of the basic branch. Later, Tao $et\ al.$ [5] propose SINT, which has two identical branches: the query branch and the search branch. The architecture is inherited from AlexNet [22]. The work in [6] proposes a fully convolutional Siamese network (SiamFC) to calculate the similarity function between two frames. RAS-Net [7] improved this similarity metric by learning the attention mechanism using a Residual Attentional Network. In [23], a semantic branch with a channel attention mechanism and an appearance branch is proposed. The semantic and appearance branches are trained separately to preserve the heterogeneity of the network. The GOTURN tracker [15] learns a generic relationship between the appearance model and the object motion. GOTURN uses a simple feed-forward network without online updating. CFNet [8] uses a Siamese network with a correlation filter as a differentiable layer. CFNet is a real-time tracker with an architecture inherited from SiamFC [6]. DSiam [24] introduces a dynamic Siamese network based on a fast transformation learning model to enable the tracker to online learn the target appearance variations. The architecture of DSiam is also inherited from SiamFC [6]. DCfnet [25] proposes a correlation layer in a lightweight Siamese network and estimates the object's location as a probability heat-map. FlowTrack [26] uses optical flow estimation, feature extraction, aggregation, and a correlation filter as special layers in the Siamese network. Moreover, a spatial temporal attention mechanism is introduced into the Siamese network for adaptive aggregation. More recently, SiamRPN [9] formulates the tracking problem as a one-shot local detection task by introducing a region proposal network after the Siamese network, which is end-to-end trained offline with large-scale image pairs. In contrast to the above networks, DensSiam [16] uses dense blocks and transition layers in a Siamese network with an attention mechanism instead of only using convoluational layers. The architecture is trained on ImageNet [22].

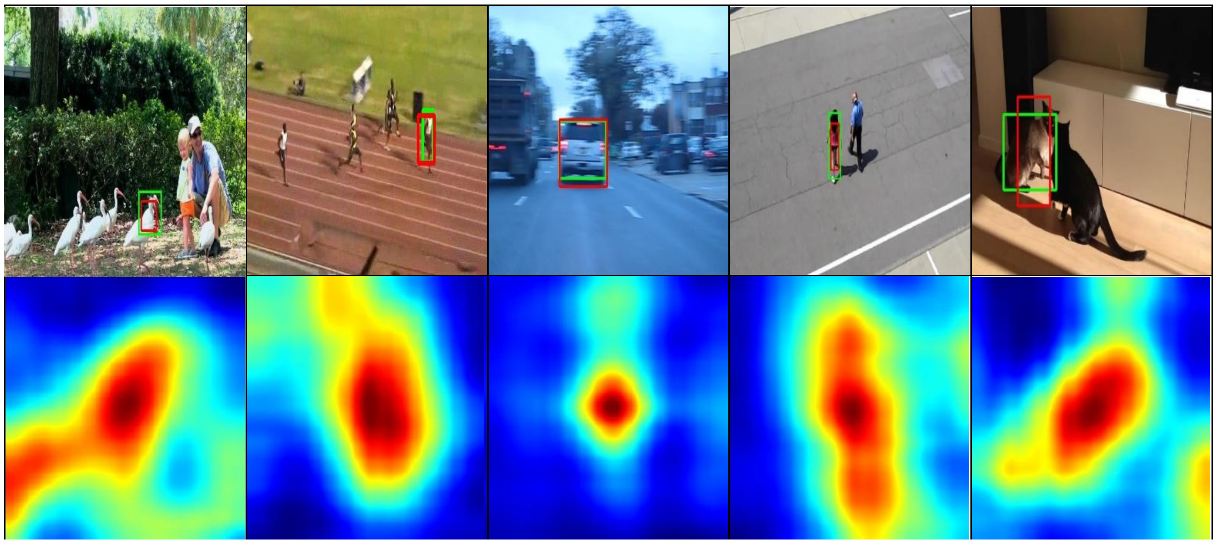
Reinforcement Learning-based Tracking: In trackers based on reinforcement learning, the objective is to learn how to take actions to maximize the reward. Rewards tell the agent how good the current perfomance is. In [27], tracking is performed based on an active detection model. The model is considered class-specific to force the agent to focus on the candidate regions and identify the correct location of the target object. In [18], the tracking is formulated as an online decision-making process, where the agent learns when to update its appearance model, reinitialize if the target has been lost, or stop updating. The tracker in [28] uses a template selection strategy to effectively choose the appropriate template for tracking based on the current frame. The selection strategy is self-learned during the training phase with randomly selected images from the dataset. The work in [29] proposes a tracker controlled by an action-decision network (ADNet). The tracker has various actions (up, down, right, left, etc.), and based on reinforcement learning, the tracker learns a good policy. The tracker in [30] uses reinforcement learning to learn a policy that decides to use either Hog features or deep features. The agent learns to select the policy based on an iterative manner. However, most of these trackers either learn a series of actions from the action pool or adaptively select the efficient features during the tracking process. Consequently, the speed and accuracy are relatively low.

5.4 Proposed Approach

5.4.1 DP-Siam Architecture

DP-Siam consists of three networks: an Agent network, an Environment network, and a Siamese network. As seen in Fig. 5.3, the input image is fed into the Agent network, which will produce an action (translation in the x and y and scale s). During offline training (indicated by the red-dashed lines in Fig. 5.3), a Q-Value guides the learning process. This is achieved by feeding the patch of the ground truth into the Environment network input layer and the Agent network action into the final layer of the Environment network. In online tracking (indicated by the green-solid lines in Fig. 5.3), the input image is fed into the Agent network to produce the predicted object's location, which is used later to feed the Environment network. The Environment network produces a score to evaluate the action taken by the agent network. Moreover, the Environment network acts as a dynamic selector for the hyper-parameters. Finally, the Siamese network produces a heat-map as shown in Fig. 5.2 with a score at each entry: the highest score indicates the location of the object being tracked. The layers' dimensions are shown in Fig. 5.3

Reinforcement learning:. Reinforcement learning is formulated using a Markov Decision Process (MDP). In MDP, the patches that contain the object of interest can be seen as states, while changes in scale s and translation (x and y) can be seen as an action. The reward can be described as a function $r(s, a)$ of a state-action pair that consists of states ($s \in S$) and actions (a). In MDP, the state transition describes how to transit from one state to another under a certain action. To adapt this definition in visual tracking, we assume that every ground truth patch or predicted bounding box is a state, and the crop operation applied on the current frame to extract the patch is a state transition to the next state s_{t+1}.



– Ground truth – DP-Siam (proposed)

Figure 5.2: The resulting heat-map of dynamic policy gradient Siamese.

5.4.2 Offline Training

DP-Siam proposes a novel Agent-Environment architecture, as depicted in Fig. 5.3, which formulates the tracking problem as a dynamic search in reinforcement learning using a Markov Decision Process (MDP). The objective is to find the policy that maximizes the cumulative discounted reward. DP-Siam handles the randomness in the formulation through maximizing the expected sum of discounted rewards. The reward function is formulated as:

$$r(s,a) = \begin{cases} +1, & \text{if } IoU >= 0.65 \\ -1, & \text{otherwise} \end{cases} \tag{5.2}$$

Note that the threshold 0.65 was chosen empirically to increase the robustness of the trackers.The network configuration during offline training is shown by the red dashed lines in Fig. 5.3. The input image is fed into the agent network to produce a three-dimensional vector (x, y, s) that represents the action. The ground truth patch is fed into the input layer of the

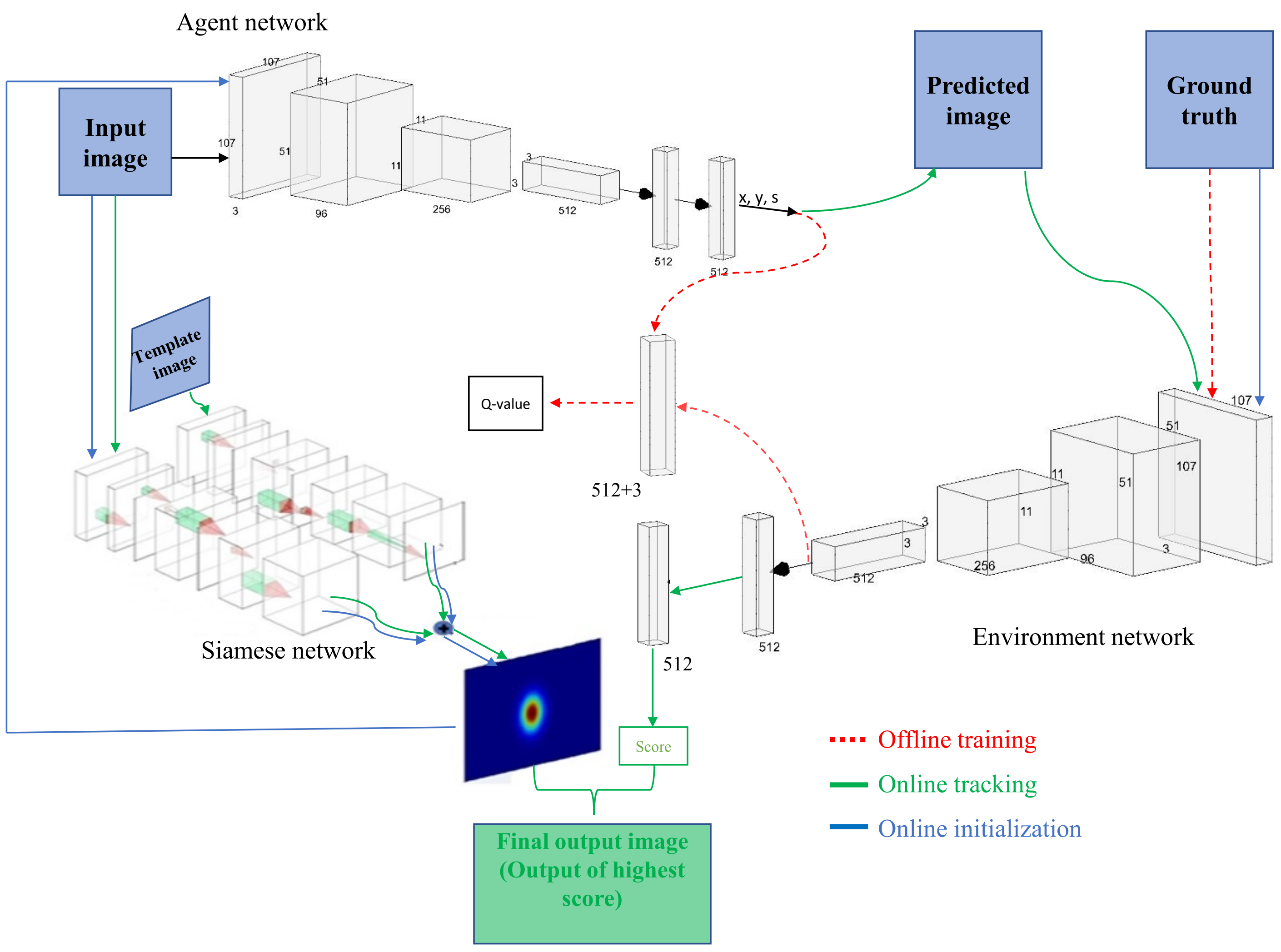


Figure 5.3: Agent-Environment architecture with Siamese network.

Environment network, while the action is fed into its final layer. The Environment network then produces a Q-value that is used to guide the learning process. In offline training, DP-Siam finds the optimal policy π^* that maximizes the expected sum of discounted rewards as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi\right] \tag{5.3}$$

where gamma γ is the discounted factor, $0 < \gamma < 1$, t is the episode length, and $\mathbb{E}(.)$ is the expectation function.

The Agent network produces an action to locate the object of interest. The action is a three-dimensional vector (x, y, and s), where x, y and s represent the horizontal translation, vertical translation, and scale changes, respectively. The Agent network works in offline training to learn a good policy to locate the object of interest. The Environment network works in offline training to produce an evaluation of the state-action pair quality (Q-value). To offline-train the Agent-Environment framework, we use the optimal Q-value to maximize the expected cumulative reward achievable from a given state-action pair as:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right] \tag{5.4}$$

where π is the policy, s_0 is the initial state, a_0 is the initial action, and Q^* is the optimal Q-value that satisfies the following Bellman equation:

$$Q^*(s,a) = \mathbb{E}_{s\prime\sim\varepsilon}\left[r + \gamma \max_{a\prime} Q^*(s\prime, a\prime) | s, a\right] \tag{5.5}$$

where $s\prime$ is the next state, $a\prime$ is the next action, and ε is the probability distribution of $s\prime$.

Eq. 5.5 is not computationally efficient because it calculates $Q^*(s,a)$ for every state-action pair. Hence, we use the Agent-Environment network to let the network learn the approximation of this equation. Consequently, the approximation of $Q^*(s,a)$ will be done using:

$$Q(s,a;\theta_A) \approx Q^*(s,a) \tag{5.6}$$

$$Q(s,a;\theta_E) \approx Q^*(s,a) \tag{5.7}$$

where θ_A and θ_E are the weights of the Agent and Environment networks, respectively, and s and a are the same state-action pair in both equations 5.6 and 5.7.

To train the DP-Siam network, the loss function L of the Agent-Environment network is formulated based on the optimal Q-value and the Expectation of the state-action pair as:

$$\begin{aligned} L_i(\theta_A; \theta_E) = \mathbb{E}_{s,a\sim p(.)} \Big[&(r + \gamma \max_{a\prime} Q(s\prime, a\prime; \theta_A) | s, a \\ &-Q(s, a; \theta_E)^2 \Big] \end{aligned} \tag{5.8}$$

s, a are sampled from the probability distribution $p(.)$ and $s\prime$, $a\prime$ are the next state and next action, respectively.

During the offline training, the framework iteratively tries to make the Q-value close to the target value. Finally, the network is updated with respect to Q as follows:

$$\nabla L_i(\theta_A; \theta_E) = \mathbb{E}_{s,a\sim p(.)} \Big[r + \gamma \max_{a\prime} Q(s\prime, a\prime; \theta_A) | s, a - Q(s, a; \theta_E) \nabla Q(s, a; \theta_E) \Big] \tag{5.9}$$

Improving offline learning: Eq. 5.9 is sufficient to be used in training with consecutive samples; however, learning from consecutive samples is challenging because the Environment network determines the next samples, which are always correlated with previous ones. Consequently, this leads to learning bias towards certain actions. To address this problem, DP-Siam proposes a novel solution based on 1) using an experience table $(s, a, r, s\prime)$ that is continually updated as episodes are running and 2) training the framework on random mini-patches of transitions from the experience table instead of using consecutive samples. This was partially inspired by Mnih $et\ al.$ [31], who used the concept of experience tables in reinforcement learning. However, to the best of our knowledge, the use of an experience table and Random Sampling of transitions have not been used in Visual Object Tracking. Moreover, we have modified the algorithm of Mnih $et\ al.$ as follows: 1) our action a_t is different, and it is a crop function, and

Algorithm 2: Training Dp-Siam with experience table

```
Initialize: Experience table $D$ to capacity $N$, Agent and Environment networks with
    random weights and $Q(s, a)$ respectively
input : Sample frames from the current sequence with its ground truth
output: Optimal weights ($\theta_A$) and ($\theta_E$)
for $episode = 1$ to $M$ do
    Select at least 20 frames randomly with their ground truth.
    for $t = 1$ to $T$ do
        With probability $\epsilon$ select an action $a_t = action[Q^*(CROP(s), a; \theta_A)]$;
        Otherwise, select a greedy action $a_t = action[\max_a Q^*(CROP(s), a; \theta_A)]$;
        Execute the action $a_t$ to observe the next state $s\prime$ and reward $r_t$;
        Set $s\prime = s$;
        Store the transitions $(s, a_t, r_t, s\prime)$ in $D$;
        Obtain normalized scores for each entry of score map using policy network ;
        Sample random mini-patch from $D$;
        Perform gradient update using Eq. 5.9;
    end
end
```

2) we normalize the output map because we have more than one network, and, hence, we have to normalize the outputs of the networks to be in the same range.

Algorithm 2 shows how to train DP-Siam with the proposed experience table. We initialize the experience table with capacity D and the Agent-Environment networks with random weights; then, for each episode, we randomly select at least 20 frames from the current sequence with their ground truth. Based on an exploration policy with probability ε, the Agent-Environment network selects an action $a_t = action[Q^*(CROP(s), a; \theta_A)]$; otherwise, it selects a greedy action $a_t = action[\max_a Q^*(CROP(s), a; \theta_A)]$. The action selection process can be further explained as follows: the network will select the action that generated this optimal Q^* and assign it to a_t. This is done by the network generating a Q^*, then the action that caused this Q^* value is assigned to a_t. This is described by the equation: $a_t = action[Q^*(CROP(s), a; \theta_A)]$. In some cases, the Q^* is still low, below the threshold ϵ. In this case, we perform a greedy action using the equation $a_t = action[\max_a Q^*(CROP(s), a; \theta_A)]$, which again finds the highest Q^* and selects the action that produced it as the value for a_t. In the implementation, we generate two lists: an optimal Q value list $(Q^* - list)$ and an actions list $(a - list)$ that generated these Q^*. Then, we apply the maximum operation to find the largest value of Q^*, and then the corresponding a from the $a - list$ is assigned to a_t. Finally, the framework is updated according to Eq. 5.9.

The crop processing is performed on the frame according to the previous state/patch and the action. Then, the new state becomes the new cropped patch, and the new reward are received from the Agent network. All transitions are stored in the experience table.

5.4.3 Online Tracking.

In online tracking, the green-solid lines in Fig. 5.3 show the interactions of the different components of the architecture while the tracking is performed. The blue lines show the interactions during online initialization. The Agent-Environment framework is initialized from two sources: 1) the ground truth in the first frame and 2) the highest 20 candidates score (patches) from the heat-map of the Siamese network. As shown in Fig. 5.3, online tracking initialization starts with the object first labelled in the first frame with its ground truth (input image). This input labelled image is fed to the Siamese network that executes a forward pass, which will produce a heat-map. The highest 20 candidates are selected from the heat-map to initialize the Agent network. Each score from these 20 candidates in the heat map is mapped into its corresponding patch from the search image through cosine interpolation [32]. The states s_t of the 20 patches in the first frame are calculated by cropping the object of interest. Finally, the Environment network is initialized with the ground truth. The Agent-Environment network is initialized from the ground truth because the actual object exists in it, and the highest 20 candidates because they will have the highest probability of the object of interest that most likely exists . It is worth mentioning that the online initialization is done only once at the beginning of the online tracking and in cases when the tracker fails ($R - measure$).

After initialization, the Agent network receives the final predicted image as an input and provides the actions a_t. The Environment network receives the predicted image from the Agent network and produces a score. At the same time, the Siamese network gives a score for each entry in the heat-map. The final object's location is determined by the highest score from the Environment network and the heat-map as shown in Algorithm 3.

5.5 Experimental Results

We trained our network on ImageNet, which means we did not include any sequences from the tested dataset. Some trackers have been trained on ImageNet such as SiamRPN, CFNET, and SiamFC. Other trackers have been trained on different datasets such as PTAV on ALOV [48], MDNET on OTB100 and VOT2014.

We used the publicly available trained networks of these algorithms and ran them as implemented by their authors on the test datasets. The results confirmed their reported results in the

```
Algorithm 3: Siamese with Agent-Environment
input  : Score of Agent-Environment $q$ and score of Siamese network $E$ (20 entries)
output: The bounding box of the tracked object
for each $E$ in Siamese heat-map do
    Normalize the Siamese score value $E$ by taking the softmax function to scale the
      values to be a probability;
    Normalize the Agent-Environment score value $q$ by taking the softmax function to
      scale the values to be a probability;
    Compare the value of $q$ and $E$;
    Choose the highest score value;
    Choose the bounding box according to the highest score value;
end
```

Table 5.1: Data dimensions in DP-Siam.

Layers	Size	Agent branch	Environment branch
$convloution$	$107 \times 107 \times 3$	$107 \times 107 \times 3$	$107 \times 107 \times 3$
$convloution$	$51 \times 51 \times 96$	$51 \times 51 \times 96$	$51 \times 51 \times 96$
$convloution$	$11 \times 11 \times 256$	$11 \times 11 \times 256$	$11 \times 11 \times 256$
$FullyConnected$	$3 \times 3 \times 512$	$3 \times 3 \times 512$	$3 \times 3 \times 512$
$FullyConnected$	$512 \times 1 \times 1$	$512 \times 1 \times 1$	$512 \times 1 \times 1$
$FullyConnected$	$512 \times 1 \times 1$	$512 \times 1 \times 1$	$512 \times 1 \times 1$

corresponding papers. We have chosen not to change anything in these algorithms to avoid bias and/or any mistakes.

Table 5.2: Comparison with the state-of-the-art trackers on VOT2015 including the top three trackers ranked by Accuracy.

Tracker	A	EAO	R	fps
DP-Siam(proposed)	0.59②	**0.39**①	0.70②	82
MDNet [11]	**0.60**①	0.38②	**0.69**①	1
OACF [33]	0.58	0.36③	1.81	5
HP[34]	0.58	0.33	1.578	69
RAJSSC [35]	0.57	0.34	1.63	22
DeepSRDCF [36]	0.56	0.32	1.05	10
EBT [37]	0.47	0.31	1.05	4
SRDCF [38]	0.56	0.29	1.24	5
BACF [39]	0.59③	-	1.56	35
EAST [30]	0.57	0.34	1.03③	159
Staple [40]	0.57	0.30	1.39	80
SiamFC [6]	0.55	0.29	1.58	86

Datasets: The proposed tracker is tested on OTB2013 [49], which has 50 sequences, as well as on OTB50 and OTB100 [50], with 50 and 100 sequences, respectively. Other experiments were conducted on UAV20L and UAV123 [51], which have 20 sequences and 123 sequences, respectively. DP-Siam is also tested on VOT2015 [52], VOT2016 [41], and VOT2018 [45], which includes all the sequences of VOT2017 as well as other sequences.

Evaluation metric: The success and precision metrics [49] are used to evaluate all trackers on OTB2013, OTB50, and OTB100. Success measures the intersection over union (IoU) of

Table 5.3: Comparison with the state-of-the-art trackers on VOT2016 including the top three trackers ranked by Accuracy.

Tracker	A	EAO	R	fps
DP-Siam(proposed)	0.540③	**0.380**①	**0.250**①	82
SSAT [41]	**0.577**①	0.321②	0.291②	22
STAPLE+ [41]	0.557②	0.286	0.368	25
SRBT [42]	0.496	0.290③	0.350	15
MDNet_N [11]	0.541	0.257	0.337③	1
HP [34]	0.539	0.242	0.46	69
DPT [43]	0.492	0.236	0.489	10
ColorKCF [41]	0.503	0.226	0.443	90
GCF [41]	0.520	0.218	0.485	7
SiamFC-A [6]	0.532	0.235	0.461	86
NSAMF [44]	0.502	0.227	0.438	25

ground truth and predicted bounding boxes. The success plot shows the rate of bounding boxes whose IoU score is larger than 20 pixels from the centre of the ground truth. The precision measures the percentage of the object location within 20 pixels from that of the ground truth. The Area Under the Curve (AUC) of success plots is applied to rank the trackers.

For the VOT2015, VOT2016, and VOT2018 datasets, we follow the Visual Object Tracking challenge (VOT) standards [49] to evaluate the tracking performance using the accuracy A (overlap with the ground truth), the expected average overlap EAO, and the robustness R (failure rate). The tracker is said to fail when there is no overlap between the predicted bounding box

Table 5.4: Comparison with the state-of-the-art trackers on VOT2018 including the top three trackers ranked by Accuracy.

Tracker	A	EAO	R	fps
DP-Siam(proposed)	**0.612**①	**0.387**①	**0.150** ①	82
SiamRPN [9]	0.586②	0.383②	0.276③	160
SA_Siam_R [23]	0.566③	0.337③	0.258②	50
FSAN [45]	0.554	0.256	0.356	30
DSiam [24]	0.215	0.196	0.646	25
HP [34]	0.331	0.2726	0.632	69
ECO [46]	0.484	0.280	0.276	60
CSRDCF [38]	0.491	0.256	0.356	13
SiamFC [6]	0.503	0.188	0.585	86
SAPKLTF [45]	0.488	0.171	0.613	25
ASMS [47]	0.494	0.169	0.623	25

Table 5.5: Effect of hyper-parameters of dynamic Siamese on the proposed tracker performance on VOT2016 and VOT2018.

Dataset	Fixed Parameters			3 parameters optimization			5 parameters optimization		
	A	EAO	R	A	EAO	R	A	EAO	R
VOT2016	0.52	0.33	0.28	0.53	0.3312	0.26	0.54	0.38	0.25
VOT2018	0.53	0.27	0.33	0.55	0.2805	0.18	0.61	0.387	0.15

Table 5.6: Comparison between the new approach (DP-Siam) and the previous authors' approach (DensSaim) on VOT2018

Tracker	A	EAO	R	fps
DensSiam [16]	0.54	0.250	0.350	60
DP-Siam (proposed)	0.61	0.387	0.150	82

Table 5.7: Ablation study using variations of DP-Siam

Variations of DP-Siam			OTB2013		OTB100		Speed
Siamese	Experience Table	Hyper-parameter Selections	AUC	Prec.	AUC	Prec.	fps
✓			0.631	0.850	0.607	0.836	$\sim$ 82
	✓		0.633	0.858	0.593	0.821	$\sim$ 82
✓	✓		0.640	0.861	0.630	0.840	$\sim$ 82
	✓	✓	0.671	0.901	0.642	0.863	$\sim$ 83
✓		✓	0.679	0.905	0.652	0.874	$\sim$ 82
✓	✓	✓	0.686	0.918	0.677	0.883	$\sim$ 82

and the ground truth. In the case of a failure, the tracker is restarted.

5.5.1 Comparison with state-of-the-art

We compare the proposed tracker (DP-Siam) with the state-of-the-art trackers on both OTB and VOT benchmarks. Conventionally, a tracking speed beyond 25 fps is considered real-time [49].

5.5.1.1 OTB benchmarks

Table 5.8 shows the comparison between DP-Siam and the state-of-the-art trackers on OTB2013,

Table 5.8: Comparison of state-of-the-art real-time trackers on OTB benchmarks.

Dataset	Measures	DP-Siam (proposed)	BACF [39]	PTAV [53]	ECOhc [46]	DSiamM [24]	EAST [30]	Staple [40]	SiamFC [6]	CFNet [8]	LMCF [54]	LCT [55]	HP [34]	AdNet [29]
OTB2013	AUC	**0.686** ①	0.656	0.663②	0.652	0.656	0.638	0.593	0.607	0.611	0.628	0.628	0.629	0.659 ③
	Prec.	**0.918** ①	0.859	0.895③	0.874	0.891	-	0.782	0.809	0.807	0.842	0.848	0.874	0.903 ②
OTB50	AUC	0.621 ②	0.570	0.581	0.592③	-	-	0.507	0.516	0.530	0.533	0.492	0.554	**0.659**①
	Prec.	0.839 ②	0.768	0.806③	0.814③	-	-	0.684	0.692	0.702	0.730	0.691	0.745	**0.898**①
OTB100	AUC	**0.677** ①	0.621	0.635③	0.643②	-	0.629	0.578	0.582	0.568	0.580	0.562	0.601	0.635
	Prec.	**0.883**①	0.822	0.849	0.856②	-	-	0.784	0.771	0.748	0.789	0.762	0.796	0.851③
	fps	82	35	25	60	25	159	80	86	75	85	27	69	3

OTB50, and OTB100. Area under curve (AUC) and precision are used to evaluate the trackers. The comparison shows that DP-Siam achieves the best performance among all trackers on the OTB2013 and OTB100 benchmarks. The comparison on OTB2013 clearly shows that DP-Siam is ranked as the best tracker in terms of AUC with a gain of 2.3% compared to the second-best tracker, which is PTAV. In terms of precision, DP-Siam is ranked as the best tracker with 0.918 compared to PTAV, which had 0.903. DP-Siam achieves the second-best performance on OTB50 with a score of 0.621 in AUC and 0.839 in precision. The best tracker in terms of AUC on OTB50 is AdNet, which has a gain of 3.8% compared to the proposed tracker. In terms of precision, AdNet is the best tracker with a gain of 5.9% compared to the proposed tracker. It should be noted that AdNet has two versions: a fast version with 15 FPS and a slow version with 3 FPS [29]. In the fast version, AdNet uses a few samples, while in the slow version, AdNet uses large samples to make an accurate decision. In Table 5.8, we used the slow version (3 FPS), which is more accurate than the fast version. The proposed tracker has one

Table 5.9: Effect of hyper-parameters selection on the proposed tracker performance on OTB2013 and OTB100.

Dataset	Fixed Parameters		3 parameters optmization		5 parameters optimization	
	AUC	Prec.	AUC	Prec.	AUC	Prec.
OTB2013	0.640	0.861	0.655	0.896	0.686	0.918
OTB100	0.630	0.840	0.657	0.865	0.677	0.883

version and works beyond real-time (82 FPS). The proposed tracker and AdNet outperform the HP tracker, which has a score of 0.629 in terms of AUC and 0.874 in terms of precision. HP tracker uses hyper-parameters optimization with different network from the proposed network. DP-Siam is ranked the first on OTB100 with 0.677 in AUC and 0.883 in precision with a gain of 3.4 % in terms of AUC. The precision plots and success plots of one path evaluation (OPE) [56] are shown in Fig. 5.4. In OTB50 and OTB100, the most dominant challenge is the occlusion; however, DP-Siam achieves a leading performance. The plots in Fig. 5.4 are obtained by running the OTB toolkit on the tracking results codes published by the authors.

Table 5.10: Summary of the results on UAV20L and UAV123.

Dataset	Measures	DP-Siam (proposed)	SiamRPN [9]	ECO [46]	MEEM [57]	SiamFC [6]	CFNet [33]	SRDCF [38]	MUSTER [15]	HP [34]	AdNet [29]
UAV20L	AUC	**0.492** ①	0.454	0.435	0.295	0.399	0.349	0.343	0.329	0.424	0.401
	Prec.	**0.681** ①	0.0.617	0.604	0.482	0.613	0.570	0.507	0.514	0.613	0.615
UAV123	AUC	0.519②	**0.527** ①	0.525	0.392	0.498	0.436	0.464	0.391	0.464	0.483
	Prec.	**0.777** ①	0.748	0.741	0.627	0.726	0.651	0.676	0.591	0.685	0.724

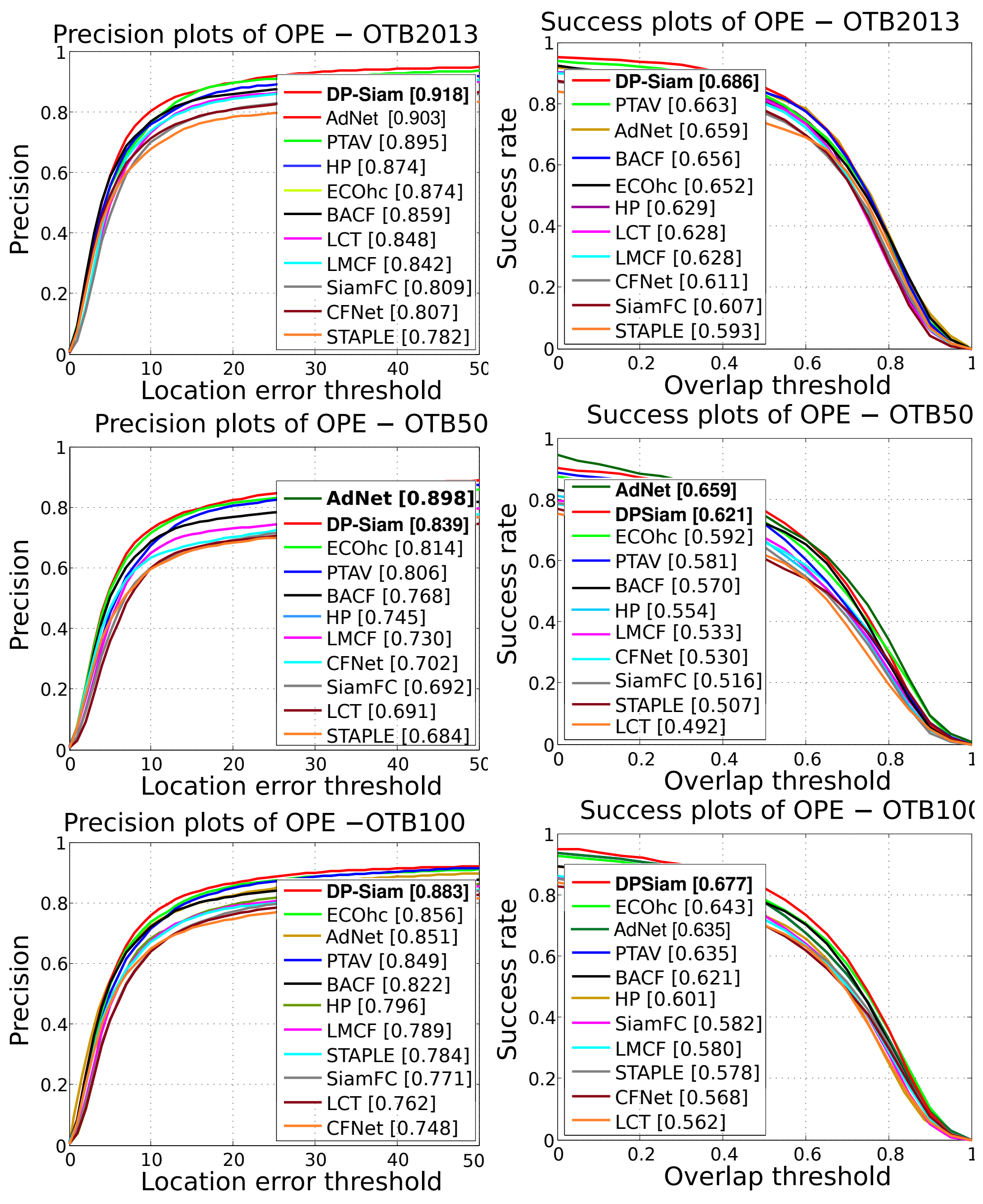


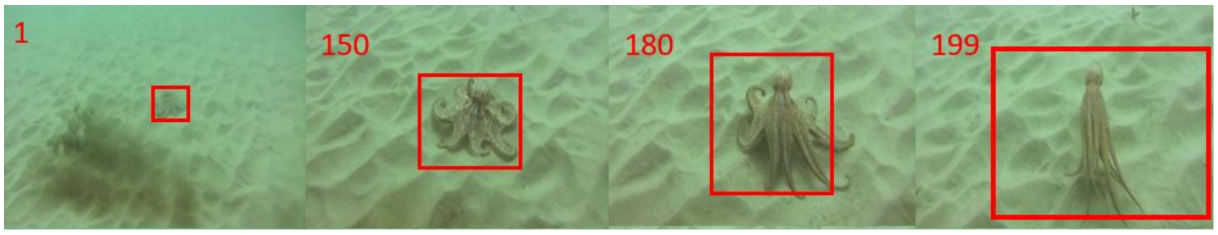
Figure 5.4: The precision plots and success plots of OTB benchmarks.

5.5.1.2 VOT benchmark

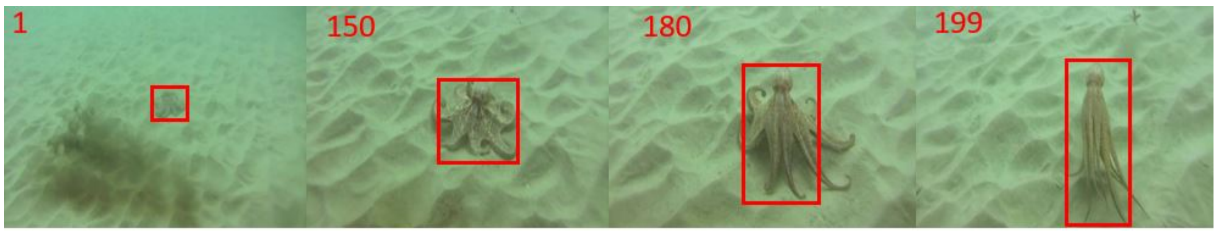
Table 5.2 shows the comparison between DP-Siam and state-of-the-art trackers on VOT2015. Raw scores generated from the VOT toolkit are listed, including the best trackers in terms of accuracy. DP-Siam is ranked as the second-best tracker in terms of accuracy with 0.59 compared to MDNet, which had an accuracy of 0.6. However, DP-Siam is ranked the best tracker on VOT2015 in the EAO measure with 0.39, followed by MDNet with 0.38. In the robustness measure, DP-Siam is the second-best tracker with 0.70 after MDNet, which had 0.69. It is obvious that DP-Siam and MDNet are the best two trackers on VOT 2015; however, DP-Siam works in real time with a speed of 82 fps compared to MDNet, which has 1 fps. Based on this, DP-Siam can be considered the overall best real-time tracker on this benchmark dataset.

Table 5.3 shows the results of the comparison on the VOT2016. DP-Siam is ranked as the third-best tracker in terms of accuracy with 0.54 compared to SSAT[41] with 0.577 and STAPLE+ [41] with 0.557. The accuracy of DP-Siam comes in third place on VOT2016 after SSAT and STAPLE+ due to a slight overfitting to some objects from the training ImageNet dataset. This problem can be solved using a dataset that has a wide variety of training objects such as a YouTube-BoundingBoxes dataset [58]. However, DP-Siam is the best tracker in the other two measures: robustness, with 0.25, and expected average overlap, with 0.38.

Table 5.4 shows the comparison of DP-Siam on VOT2018. DP-Siam is ranked as the best tracker in terms of accuracy with 0.612 and gain over 2.6 % compared to the second-best tracker, which is Siam-RPN. DP-Siam is also ranked as the best tracker in terms of robustness, with 0.150, and expected average overlap, with 0.387. It is worth noting that DP-Siam outperforms SiamRPN (the best tracker in the VOT2018 challenge) with a gain over 12.6% in robustness. It is evident from Table 5.4 that DP-Siam also outperforms all top trackers in the VOT2018 challenge in all three evaluation measures. It should be noted that AdNet has been trained on VOT2013, Vot2014, VOT2015, and ALOV300 [59] benchmarks, and many sequences from these benchmarks already exist on VOT2016 and VOT2018 [29]. Consequently, comparing our tracker or other state-of-the-art trackers to AdNet on the VOT benchmark is not objective and will produce biased results in favor of AdNet. More qualitative results are given in Fig. 5.9.



(a)

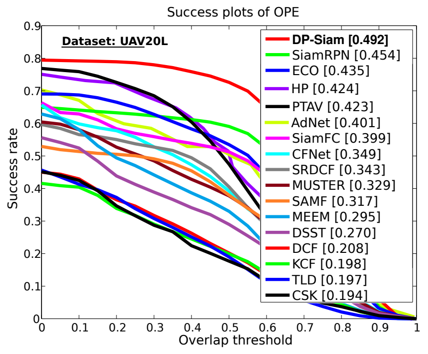


(b)

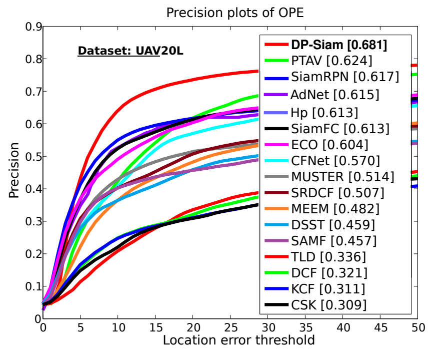
Figure 5.5: DP-Siam results on Octopus sequence with: (a) Fixed hyper-parameters and (b) Dynamic hyper-parameters.

5.5.1.3 UAV Dataset

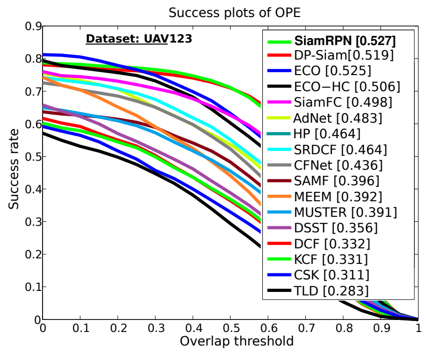
UAV [51] contains a long-term evaluation subset (UAV20L) and a short-term evaluation subset (UAV123). The evaluation is done using precision and success plots. DP-Siam is compared against the state-of-the-art trackers for UAV. The comparison includes: ECO [46], PTAV [53] , SiamRPN [9], SiamFC [6] and CFNet [46] as shown in Fig. 5.6. In Fig. 5.6 (a) the success



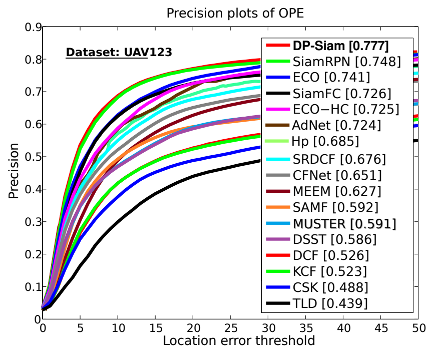
(a)



(b)



(c)

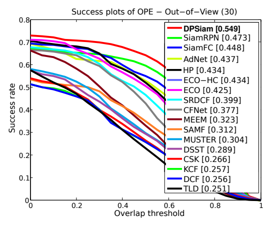


(d)

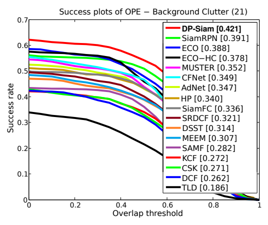
Figure 5.6: Summary of the results on UAV, (a) and (b) show the success and precision plots on UAV20L, (c) and (d) show the success and precision plots on UAV123.

plot shows that DP-Siam outperforms SiamRPN, as DP-Siam has a success of 0.492, while SiamRPN has a success of 0.454. Similarly, the precision plot Fig. 5.6 (b) shows that DP-Siam outperforms PTAV and SiamRPN. We conducted another experiment on the UAV123 dataset as shown in Fig. 5.6 (c) and (d). In Fig. 5.6 (c) the success plot shows that SiamRPN outperforms DP-Siam with a gain of 0.8%. However, in Fig. 5.6 (d), DP-Siam outperforms SiamRPN with a gain of 2.9% and ECO with 3.6%.

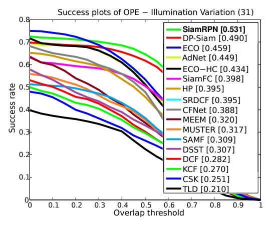
Individual Challenging Attributes: To study the performance of the proposed tracker un-



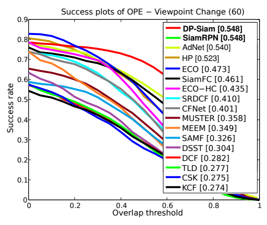
(a)



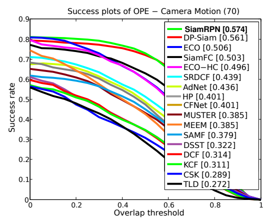
(b)



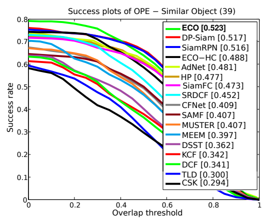
(c)



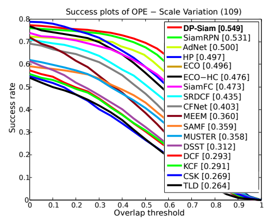
(d)



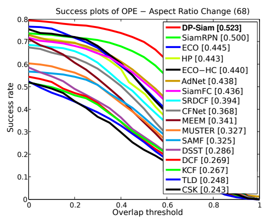
(e)



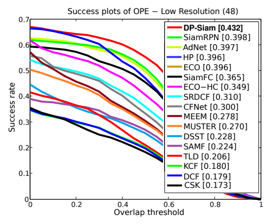
(f)



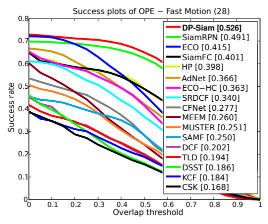
(g)



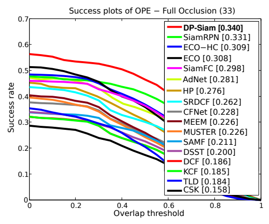
(h)



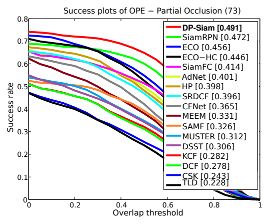
(i)



(j)

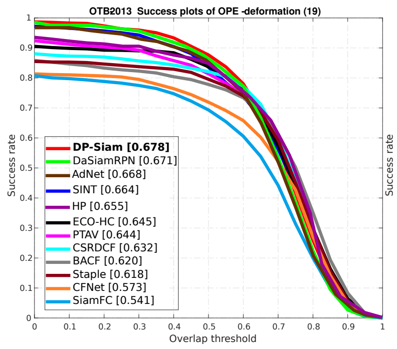


(k)

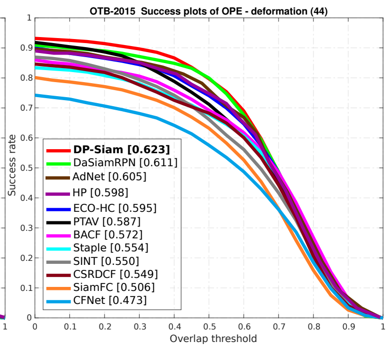


(l)

Figure 5.7: Success plots with 12 attributes on UAV123.



(a)

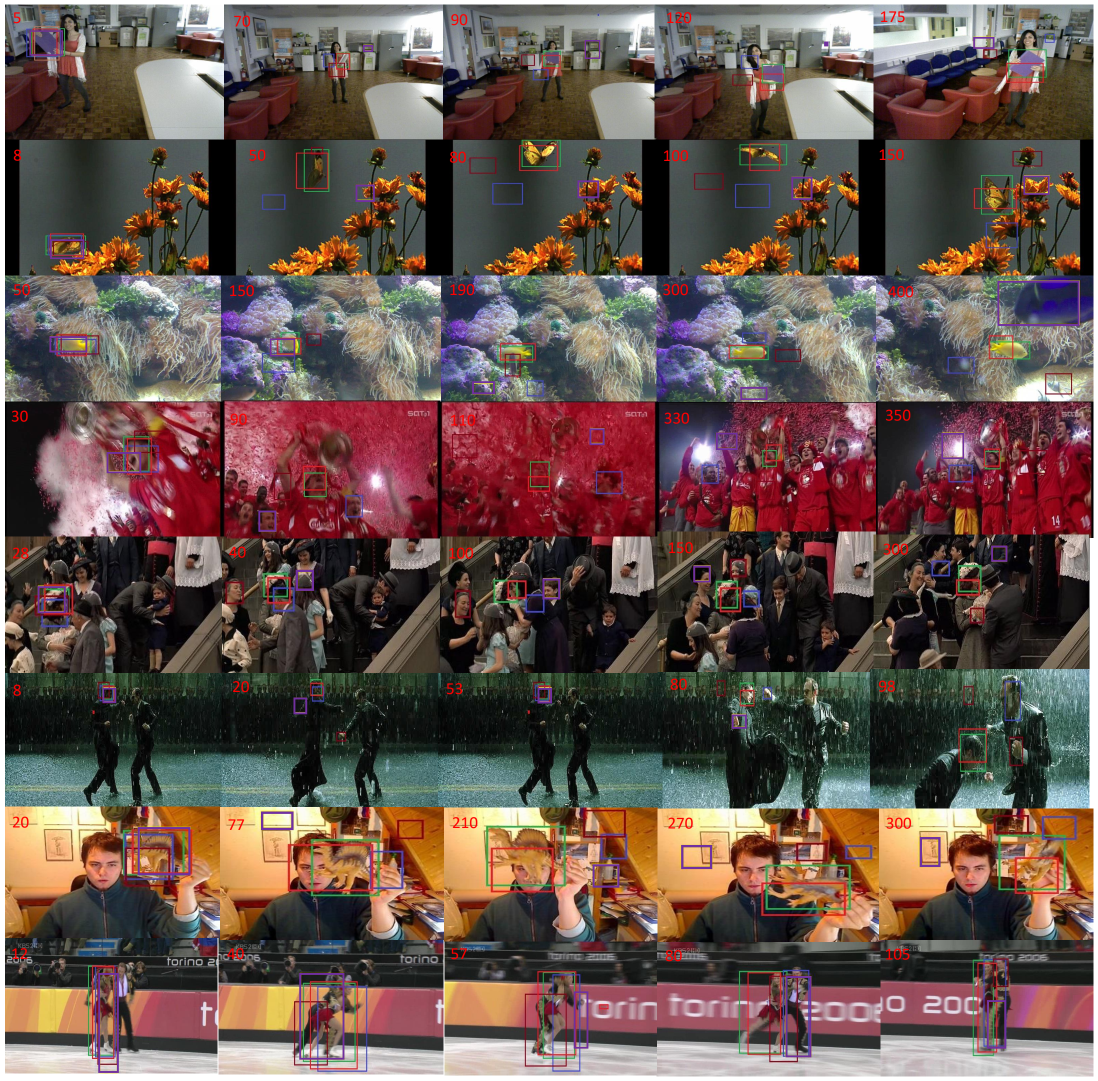


(b)

Figure 5.8: Success plots on:(a) deformation on OTB2013 and (b) deformation on OTB2015

der different individual circumstances, another experiment was conducted on 12 challenging attributes in the UAV123 dataset, including out-of-view, background clutter, illumination variation, viewpoint change, camera motion, similar object, scale variation, aspect ratio change, low resolution, fast motion, full occlusion, and partial occlusion. The success plots' results are shown in Fig. 5.7. The plots clearly show that the proposed tracker outperforms the state-of-the-art trackers in 10 out of the 12 attributes, and places in second in the remaining two attributes. Table 5.10 summarizes the results on UAV201L and UAV123.

Moreover, another important attribute, deformation, was not included in the UAV datasets. To evaluate the proposed algorithm in regard to the deformation of the object being tracked, we evaluated DP-Siam on the deformation sequences (e.g. Basketball, Bird1, and Bird2) in OTB2013 and OTB2015. Fig. 5.8 shows that DP-Siam outperforms the state-of-the-art trackers in the deformation challenge.



Ground truth – DP-Siam (Ours) – SiamFC – SRDCF – KCF

Figure 5.9: Visual results of the proposed method on challenging sequences from VOT2018 dataset. The rows show different sequences, while the columns show the trackers output over time (the frame number is printed on the top left corner of the image. The sequences in vertical order, *book, butterfly, fish3, soccer1, godfather, matrix, dinosaur* and *iceskater2*.

5.5.2 Effect of Dynamic Hyper-parameters

Dynamically selecting the hyper-parameters allows the tracker to adapt to significant scale changes that affect the object's appearance model, and, hence, allow it to have better performance. If the hyper-parameters (scale penalty, scale learning rate, window weight, and template learning rate) are fixed, the tracker will have a better performance on some sequences than on other sequences. To show the effect of the dynamic hyper-parameters selection, consider the sequence 'Octopus'. In the case of dynamic hyper-parameter selection, the bounding box will appear as shown in Fig. 5.5 (a). However, in a case using dynamic hyper-parameters selection, the bounding box will appear as shown in Fig. 5.5 (b). We performed an experiment to show the effect of including/excluding dynamic hyper-parameters selection in Table 5.7. This experiment clearly shows that by adding the hyper-parameters' components, it increases the tracker performance.

During training, we initialize the framework with initial values for the five hyper-parameters from random distribution; then, the framework iteratively updates these hyper-parameters to minimize the loss function. During online tracking, the network has learned the best values for the five hyper-parameters during the offline training, and, hence, it will select the best values during the online tracking to minimize the loss function.

To show the effect of using dynamic hyper-parameters selection versus using fixed hyper-parameters, three experiments were conducted using the OTB2013, OTB50, and OTB100 as follows: 1) fixing all the hyper-parameters, 2) three hyper-parameters are dynamically selected and 3) five hyper-parameters are dynamically selected. The results in Table 5.9 show that dynamic hyper-parameters selection provides better performance since they can accommodate more challenges. DP-Siam has five hyper-parameters (scale step, scale penalty, scale learning rate, window weight, and template learning rate). If the hyper-parameters are being fixed along the entire sequence, DP-Siam will work well within these hyper-parameters. On the other hand, if the hyper-parameters are dynamically selected, the bounding box around the object of interest will adapt to the shrinking and expansion.

The Environment network also acts as a dynamic selector for the hyper-parameters. We conducted three experiments on dynamically selecting the hyper-parameters: 1) fixing the five hyper-parameters, 2) the first three hyper-parameters (scale step, scale penalty and scale learning rate) are dynamically selected, and 3) the five hyper-parameters are dynamically selected.

Table 5.5 shows the effect of dynamic hyper-parameters on the tracking performance in the case of fixing hyper-parameters and dynamic hyper-parameters. The comparison includes three hyper-parameters optimization and five hyper-parameters optimization. DP-Siam is acting poorly in the case of fixing the hyper-parameters because the fixed hyper-parameters might be suitable for particular sequences while not suitable for others. The results in the case of dynamic hyper-parameters are better because dynamic hyper-parameters accommodate more challenges. The results of five hyper-parameters optimization are more accurate than the three hyper-parameters because more hyper-parameters accommodate more challenges.

5.5.3 Comparison with DensSiam

Experiments were conducted to compare the proposed tracker and the authors' previous work, DensSiam [16], as shown in Table 5.6. The new proposed architecture in DP-Siam significantly increases the accuracy and expected average overlap. DP-Siam is trained to dynamically select the policy and hyper-parameters in contrast to DensSiam and most other Siamese trackers that assign fixed hyper-parameters for the whole dataset.

5.5.4 Ablation Study

Table 5.7 presents an ablation study that highlights some variations of DP-Siam. Three variations of DP-Siam were tested and evaluated on OTB2013 and OTB100. The baseline ”Siamese” is first evaluated, and it uses a Densely-Siamese architecture. ”Experience Table” is used as a memory for DP-Siam to store the Q-value, while ”Hyper-parameter selections” is used with five hyper-parameters to automatically select the most suitable hyper-parameters.
Table 5.7 shows the importance of each component. With only the ”Siamese” network, DP-Siam gives a poor performance. Similarly, DP-Siam with only ”Experience Table” or ”Hyper-parameter selection” gives low performance. Adding compilations of the components substantially increases the DP-Siam performance. It is worth mentioning that adding all components to DP-Siam gives the best performance in terms of accuracy and precision.

5.5.5 Implementation details

We used the ImageNet Large Scale Visual Recognition Challenge (ILSVRC15) [60] for the training of DP-Siam. The ILSVRC15 contains 1.3 million labelled frames in 4000 sequences, and it has a wide variety of objects that contribute to the generalization of DP-Siam.

Hyper-parameters initial settings: Training is performed over 100 epochs, each with 53,200 sampled pairs. Stochastic gradient descent (SGD) is applied with a momentum of 0.9 to train the network. We adopt the mini-patches of size 8, and the learning rate is annealed geometrically at each epoch from 10^{-3} to 10^{-8}. We implement DP-Siam in TensorFlow [61] 1.8 framework. The experiments are performed on a PC with a Xeon E5 2.20 GHz CPU and a Titan XP GPU.

Tracking settings: We adapt the initial scale variations of O^s where $O = 1.0375$ and $s =$

$\{-2, 0, 2\}$. The input target image size is 127×127 and the search image size is 255×255. We use linear interpolation to update the scale with a factor of 0.764.

5.6 Conclusions and Future Work

In this chapter, a novel dynamic Siamese tracker is proposed with a policy gradient to dynamically train the policy to select the best action from the action space. Unlike traditional Siamese trackers, which have fixed hyper-parameters for the entire dataset, the proposed architecture dynamically selects the hyper-parameters for each frame or group of frames. The proposed tracker uses reinforcement learning to learn and predict the next hyper-parameters set based on the gradient policy. The proposed tracker substantially increases the generalization and the expected average overlap since the bounding box can be precisely adapted with the shape of objects. Future work includes extending DP-Siamese to use a Bayesian inference network to predict more hyper-parameters. Future work also includes training DP-Siam with a YouTube-BoundingBoxes dataset.

References

[1] K. Lenc and A. Vedaldi, “Understanding image representations by measuring their equivariance and equivalence,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 991–999.

[2] A. Kendall, M. Grimes, and R. Cipolla, “Convolutional networks for real-time 6-dof camera relocalization. corr abs/1505.07427 (2015),” 2015.

[3] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, “Online detection and

classification of dynamic hand gestures with recurrent 3d convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4207–4215.

[4] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[5] R. Tao, E. Gavves, and A. W. Smeulders, "Siamese instance search for tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1420–1429.

[6] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*. Springer, 2016, pp. 850–865.

[7] Q. Wang, Z. Teng, J. Xing, J. Gao, W. Hu, and S. Maybank, "Learning attentions: residual attentional siamese network for high performance online visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4854–4863.

[8] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end representation learning for correlation filter based tracking," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5000–5008.

[9] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8971–8980.

[10] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[11] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[12] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” in *International Conference on Learning and Intelligent Optimization*. Springer, 2011, pp. 507–523.

[13] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[14] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[15] D. Held, S. Thrun, and S. Savarese, “Learning to track at 100 fps with deep regression networks,” in *European Conference on Computer Vision*. Springer, 2016, pp. 749–765.

[16] M. H. Abdelpakey, M. S. Shehata, and M. M. Mohamed, “Denssiam: End-to-end densely-siamese network with self-attention model for object tracking,” in *International Symposium on Visual Computing*. Springer, 2018, pp. 463–473.

[17] A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, “Robust 3d object track-

ing from monocular images using stable parts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1465–1479, 2018.

[18] J. S. Supancic III and D. Ramanan, "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning." in *ICCV*, 2017, pp. 322–331.

[19] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang, "Deep reinforcement learning for visual object tracking in videos," *arXiv preprint arXiv:1701.08936*, 2017.

[20] R. Yu, I. Cheng, B. Zhu, S. Bedmutha, and A. Basu, "Adaptive resolution optimization and tracklet reliability assessment for efficient multi-object tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 7, pp. 1623–1633, 2018.

[21] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[23] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4834–4843.

[24] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *The IEEE International Conference on Computer Vision (ICCV).(Oct 2017)*, 2017.

[25] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu, “Dcfnet: Discriminant correlation filters network for visual tracking,” *arXiv preprint arXiv:1704.04057*, 2017.

[26] Z. Zhu, W. Wu, W. Zou, and J. Yan, “End-to-end flow correlation tracking with spatial-temporal attention,” *illumination*, vol. 42, p. 20, 2017.

[27] J. C. Caicedo and S. Lazebnik, “Active object localization with deep reinforcement learning,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2488–2496.

[28] J. Choi, J. Kwon, and K. M. Lee, “Visual tracking by reinforced decision making,” *arXiv preprint arXiv:1702.06291*, 2017.

[29] S. Y. J. C. Y. Yoo, K. Yun, and J. Y. Choi, “Action-decision networks for visual tracking with deep reinforcement learning,” 2017.

[30] C. Huang, S. Lucey, and D. Ramanan, “Learning policies for adaptive tracking with deep feature cascades,” in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 105–114.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[32] J. Agbinya, “Interpolation using the discrete cosine transform,” *Electronics letters*, vol. 28, no. 20, pp. 1927–1928, 1992.

[33] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, “The importance of estimating object extent when tracking with correlation filters,” *Preprint*, 2015.

[34] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, and F. Porikli, “Hyperparameter optimization for tracking with continuous deep q-learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 518–527.

[35] M. Zhang, J. Xing, J. Gao, X. Shi, Q. Wang, and W. Hu, “Joint scale-spatial correlation tracking with adaptive rotation estimation,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 32–40.

[36] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, “Convolutional features for correlation filter based visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 58–66.

[37] G. Zhu, F. Porikli, and H. Li, “Tracking randomly moving objects on edge box proposals,” *arXiv preprint arXiv:1507.08085*, 2015.

[38] A. Lukezic, T. Vojir, L. C. Zajc, J. Matas, and M. Kristan, “Discriminative correlation filter with channel and spatial reliability.” in *CVPR*, vol. 6, 2017, p. 8.

[39] H. K. Galoogahi, A. Fagg, and S. Lucey, “Learning background-aware correlation filters for visual tracking.” in *ICCV*, 2017, pp. 1144–1152.

[40] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, “Staple: Complementary learners for real-time tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1401–1409.

[41] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, “The visual object tracking vot2015 challenge

results," in *Proceedings of the IEEE international conference on computer vision workshops*, 2016, pp. 1–23.

[42] H. Lee, Alex, and K. Daijin, "Salient region-based online object tracking," in *IEEE Winter Conference on Applications of Computer Vision*, 2017, pp. 1170–1177.

[43] O. Akin, E. Erdem, A. Erdem, and K. Mikolajczyk, "Deformable part-based tracking by coupled global and local correlation filters," *Journal of Visual Communication and Image Representation*, vol. 38, pp. 763–774, 2016.

[44] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *European conference on computer vision*. Springer, 2014, pp. 254–265.

[45] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pfugfelder, L. C. Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey, G. Fernandez, and et al., "The sixth visual object tracking vot2018 challenge results," 2018.

[46] M. Danelljan, G. Bhat, F. S. Khan, M. Felsberg *et al.*, "Eco: Efficient convolution operators for tracking." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[47] T. Vojir, J. Noskova, and J. Matas, "Robust scale-adaptive mean-shift for tracking," *Pattern Recognition Letters*, vol. 49, pp. 250–258, 2014.

[48] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1442–1468, 2014.

[49] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

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

[51] M. Mueller, N. Smith, and B. Ghanem, “A benchmark and simulator for uav tracking,” in *European conference on computer vision*. Springer, 2016, pp. 445–461.

[52] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, “The visual object tracking vot2015 challenge results,” in *Proceedings of the IEEE international conference on computer vision workshops*, 2015, pp. 1–23.

[53] H. Fan and H. Ling, “Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking,” in *Proc. IEEE Int. Conf. Computer Vision, Venice, Italy*, 2017.

[54] M. Wang, Y. Liu, and Z. Huang, “Large margin object tracking with circulant feature maps,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA*, 2017, pp. 21–26.

[55] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, “Long-term correlation tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5388–5396.

[56] N. Wang and D.-Y. Yeung, “Learning a deep compact image representation for visual tracking,” in *Advances in neural information processing systems*, 2013, pp. 809–817.

[57] J. Zhang, S. Ma, and S. Sclaroff, “Meem: robust tracking via multiple experts using entropy minimization,” in *European conference on computer vision*. Springer, 2014, pp. 188–203.

[58] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke, "Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on.* IEEE, 2017, pp. 7464–7473.

[59] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1442–1468, 2013.

[60] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[61] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: large-scale machine learning on heterogeneous distributed systems. arxiv preprint (2016)," *arXiv preprint arXiv:1603.04467*.

Chapter 6

NullSpaceNet: Nullspace Convoluional Neural Network with Differentiable Loss Function

6.1 abstract

We propose NullSpaceNet, a novel network that maps from the pixel-level image to a joint-nullspace (as opposed to the traditional feature space), where the newly learned joint-nullspace features have a clear interpretation and are more separable. NullSpaceNet ensures that all input images that belong to the same class are collapsed into one point in this new joint-nullspace, and the different classes are collapsed into different points with high separation margins. Moreover, a novel differentiable loss function is proposed that has a closed-form solution with no free-parameters. NullSpaceNet's architecture consists of: 1) a feature extractor backbone (i.e., the convolution and pooling layers), which is used to extract features from the input, and 2) a

nullspace layer, which maps from the pixel-level image to the joint-nullspace. Consequently, a reduction in learnable parameters has been achieved. NullSpaceNet is architecture-agnostic, which means that it can use any feature extractor as a backbone in its first components. NullSpaceNet exhibits superior performance when tested over 4 different datasets against VGG16, MobileNet-224, and MNASNET1-0. In general, NullSpaceNet needs only around $1 - 30\%$ of the time it takes a traditional CNN to classify a batch of images, and with better accuracy of up to 2.57%.

6.2 Introduction

In recent years, Convolutional Neural Networks (CNNs) have revolutionized computer vision tasks such as object tracking [1–4], surveillance systems [5], image understanding [6], computer interactions [7] and generative models [8]. Image classification is one of the core tasks in computer vision, especially in Large Scale Visual Recognition Challenges (e.g., ILSVRC15) [9]. Most classification networks consist of two components: 1) the feature extractor and 2) the classifier. The feature extractor uses a stack of convolutional layers to extract the deep features from the input images through consecutive convolutional operations. The classifier uses fully-connected layers with a softmax layer. It has been proven that most learnable parameters within the classification network are located in the fully connected layers [10]. For example, the classifier in VGG16 has 102.76 million parameters, while the feature extractor has only 32 million parameters. Consequently, this huge amount of learnable parameters requires extensive computations during both training and inference.

In this chapter, we propose NullSpaceNet, a novel network that maps from the pixel-level image to a joint-nullspace, as opposed to a traditional CNN that maps to a classical feature space. The newly learned nullspace features have a clear interpretation and are more separable. In particular, instead of using the fully connected layers with the categorical cross-entropy, NullSpaceNet maps the pixel-level image to a joint-nullspace. All input images from the same class are collapsed into one point in this new joint-nullspace and the different classes are collapsed into different points with high separation margins. Moreover, the hyperplane that has the orthonormal vectors of the projected nullspace features is well-defined and can be described as shown in Fig. 6.4 and Eq. 6.21.

The architecture of NullSpaceNet consists of: 1) a feature extractor backbone (i.e., the convolution and pooling layers), which is used to extract features from the input, and 2) a nullspace layer, which maps from the pixel-level image to the joint-nullspace. NullSpaceNet is architecture-agnostic, which means that it can use any feature extractor as a backbone in its first component. For example, the VGG16 feature extractor component (i.e., VGG16 without the fully connected layers) can be used as the feature extractor backbone of NullSpaceNet. The core idea of NullSpaceNet is to minimize the within-class scatter matrix to be zero or very close to zero, while maintaining the between-class scatter matrix to be always positive. This makes the classification task more robust as shown in Fig. 6.5. The pre-trained network and the source code are available online. [1]

To summarize, the main contributions of this chapter are:

1. A novel Network (NullSpaceNet) that learns to map from the pixel-level image to a joint-nullspace. The formulation of NullSpaceNet ensures that the nullspace features from the same class are collapsed into a single point while the ones from different classes are collapsed into different points with high separation margins. NullSpaceNet is architecture-agnostic, which means that it can easily integrate different feature extractors in its archi-

[1] https://github.com/NullSpaceNet

tecture.

2. A differentiable loss function is developed to train NullSpaceNet. The proposed loss function is different from the standard categorical cross-entropy functions. The proposed loss function ensures that the within-class scatter matrix vanishes while maintaining a positive between-class scatter matrix. The differentiable loss function has a closed-form solution with no free-parameters.

3. The proposed NullSpaceNet has a clear interpretation of the learned features, both mathematically and geometrically.

These three contributions resulted in NullSpaceNet needing only around $1 - 30\%$ of the time it takes a traditional CNN to classify a batch of images, and with a better accuracy of up to 2.57% over all 4 datasets we used in testing.

The rest of the chapter is organized as follows: Related work is presented in section 6.3, then section 6.4 details the proposed NullSpaceNet. The training and inference phases are presented in section 6.5. The experimental results are presented in section 6.6. The results and discussion are detailed in 6.7. Finally, section 6.8 concludes the chapter.

6.3 Related Work

Nullspace and Linear Discriminant Analysis (LDA) have existed as analytical methods for a significant period of time [11–15], and[16]. LDA has frequently been employed as a dimensionality reduction tool or feature extractor within the field of classification [17–20, 20–24] , and [25]. Nullspace can be derived from the Fisher-criterion objective function in an analytical way. The work in [26] used multiple local nullspaces to detect the small moving objects in aerial videos. Using the nullspace allows the detector to nullify the background while maintaining the moving objects.

Nullspace has been used in [27] to specify whether the incoming data belongs to the existing class or not. In particular, they used Incremental Kernel Nullspace Discriminative (IKNDA). To speed up their method, an intelligent update scheme is used to extract information from newly added samples. The work in [28] proposed Max-Mahalanobis distribution (MMD) using LDA to improve the robustness of the adversarial attack. [29] proposed to learn a capsule subspace using orthogonal projection. The length of the resultant capsules is utilized to score the probability of belonging to different categories. The authors in [30] proposed application of Hybrid Orthogonal Projection Estimation (HOPE) to CNN for image classification. HOPE is a hybrid model that combines orthogonal linear projection, for feature extraction, with mixture models. The idea in HOPE is to allow for extraction of useful information from high-dimension feature vectors while filtering out irrelevant noise. [31] used LDA with the Fisher-criterion on VGG16 to classify facial gender. LDA was applied on the output of the last layer to derive a light weight version of VGG16. A Bayesian classification is then used to classify the output. Notice that this is completely different from NullSpaceNet, where we reformulate the learning process in a differentiable way to train the network to learn a joint-nullspace. DeepLDA [32] proposed to use LDA to learn to maximize Eigenvalues of the Fisher-criterion. After training, DeepLDA uses the entire training set to extract the dominant basis vectors to project the new samples. In contrast to all previous methods, we use the nullspace in VGG16 in a learnable way with a differentiable loss function to project the pixel-level image to a joint-nullspace.

The only work we found that included using LDA in a deep learning framework was presented in [32], where the authors solved the LDA and integrated it in a deep CNN. It is worth mentioning that the work in [32] did not include any reference to any usage of nullspace. In our work, we do not solve for the LDA, instead we reformulate the problem within the nullspace to train the network to project from the pixel-level image onto the joint-nullspace.

6.4 NullSpaceNet

The formulation of NullSpaceNet can be applied to different feature extractor backbones. In this section, for the sake of demonstration, the formulation is applied to NullSpaceNet that uses the VGG16 feature extractor as its backbone in the first component of the network. This formulation is also applied to other feature extractor backbones as outlines in section 6.4.6.

6.4.1 Problem Definition

Given a dataset of training images $X = \{x_1, x_2, ..., x_N\} \in \mathbb{R}^{w \times h \times d}$, where w, h and d are the width, height, and depth of each image, respectively, and N is the number of images in the training dataset. Each image is associated with a respective class C, where $C = \{c_1, c_2, ..., c_n\} \in \mathbb{R}$ and n is the number of classes in the training dataset. In this chapter, the VGG16 feature extractor component $\phi(x; \theta)$ is used as the backbone feature extractor in NullSpaceNet.

The objective is to force the network to learn a joint-nullspace that maps from the pixel-level image to a strong discriminative nullspace. The learned nullspace will replace the classifier component, which has the most learnable parameters within the network. In other words, there will be no fully connected layers, but instead a learned joint-nullspace as shown in Fig. 6.1.

6.4.2 Proposed Architecture

NullSpaceNet's feature extractor architecture uses the VGG16 feature extractor as a backbone. With these settings, NullSpaceNet has 19 layers, each layer consists of (Conv-BatchNormalization-ReLu), the pooling is considered as a stand-alone layer. A (Conv-BatchNormalization-ReLu) layer can be added before the nullspace layer to accommodate different backbones and datasets. The novelty of NullSpaceNet lies in the nullspace layer and the differentiable loss function, which is detailed in section 6.4.3. The nulllspace layer forces the network, through backpropagation, to learn the projection from the pixel-level image to a joint-nullspace, where the joint-nullspace features have optimal separation margins. The Nullspace layer achieves this through spanning vectors of the optimal within-class scatter matrix. This will be discussed in more detail in section 6.4.3. Formulating the nullspace layer in this way prevents the network from the Small Sample Size (SSS) problem (i.e., the model has a high dimensional output features while training on small batches of images).

6.4.3 Mathematical Formulation of The Loss Function

Background: To derive a differentiable loss function to train the joint-nullspace, we start from the linear discriminant analysis (LDA) [33]. In this chapter, we assume that the output of the feature extractor component in the network for each batch is $F \in \mathbb{R}^{D\times N}$, where D is the dimension of the output of the feature extractor and N is the number of images. The objective of the LDA is to find a projection matrix $P \in \mathbb{R}^{d\times N}$, where $d < D$, that minimizes the within-class scatter matrix and maximizes the between-class scatter matrix simultaneously. This can

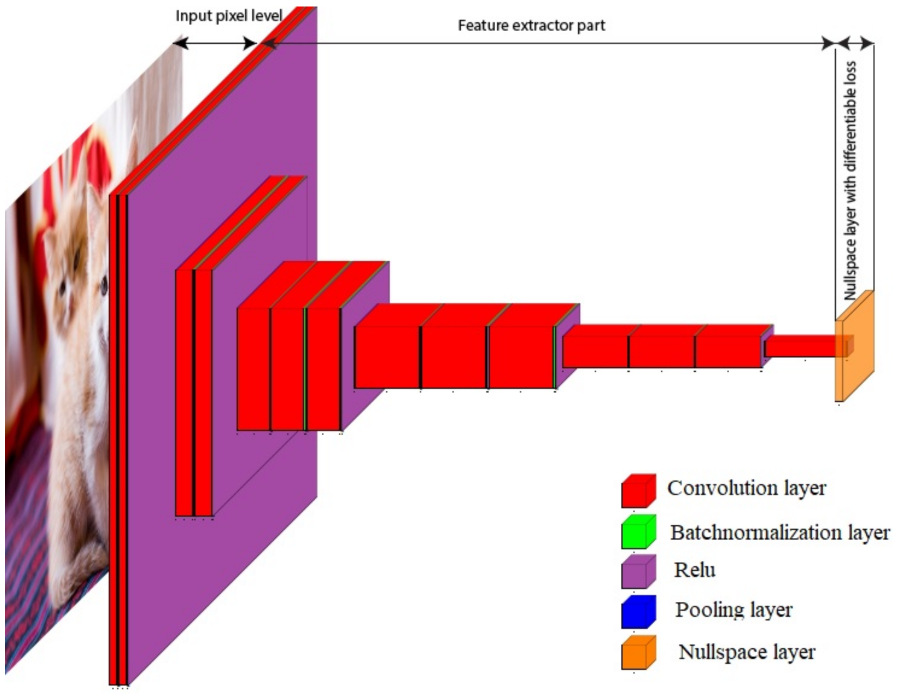


Figure 6.1: NullSpaceNet architecture,$VGG16$ feature extractor component is used as a backbone. The NullSpace layer is at the end of the architecture. Note that the nullspace layer has been magnified for the sake of visualization.

be achieved by maximizing the Fisher-discriminant criterion $\mathcal{J}(P)$ as follows:

$$\mathcal{J}(P) = \frac{P^\top S_b P}{P^\top S_w P} \tag{6.1}$$

Where P is the projection matrix, S_b and S_w are the between-class and within-class scatter matrices, respectively. The optimization of Eq. 6.1 can be solved for the generalized Eigenvalue problem as follows:

$$S_b P = \lambda S_w P \tag{6.2}$$

where λ is the Eigenvalue of the eigenvector $S_w^{-1}S_b$.

Derivation of the proposed novel loss function:

Lemma 1: The traditional VGG16 with FC has a tendency to minimize the within-class scatter matrix, however, it does not put constraints on the between-class scatter matrix.

Proof: From the visualization in Fig. 6.5 (b) and using visual inspection, the learned features of VGG16 with FC layers are scattered with no constraints on the between-class scattered matrix; for example, some classes (e.g., class #4, #5 and class #2 and #8) overlap.

Using Lemma 1 in NullSpaceNet: In NullSpaceNet, we force two constraints on the learning process. In particular, we force the between-class scatter matrix to always be positive while minimizing the within-class scatter matrix to be zero as follows:

$$P^{\top} S_b P > 0 \tag{6.3}$$

$$P^{\top} S_w P = 0 \tag{6.4}$$

Lemma 2: When NullSpaceNet satisfies the two constraints from Lemma 1 (Eq. 6.3 and 6.4), the distribution of the same class features in the new joint-nullspace approaches the Dirac Delta function.

Proof: Assuming the features are represented by the normal distribution, for simplicity, we will use a 1-D normal distribution.

$$\int g(\bar{x}) f\left(\bar{x} | \bar{\mu}, \bar{\sigma}^2\right) d\bar{x} \tag{6.5}$$

where $g(\bar{x})$ is the mean value function of the projected features by the network to the joint-nullspace, $\bar{\mu}$ is Gaussian mean, and $\bar{\sigma}$ is the standard deviation of the distribution. We take the

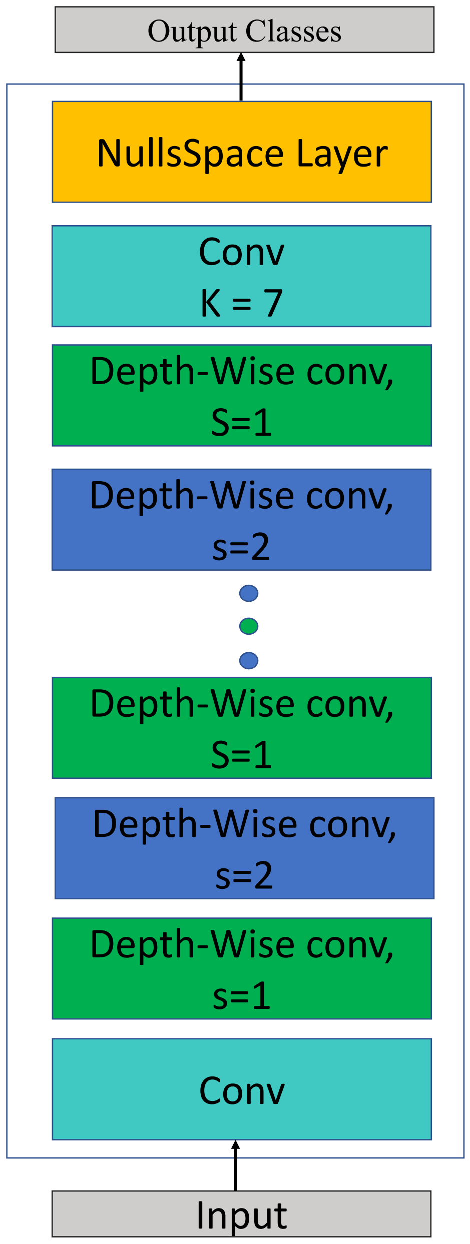


Figure 6.2: NullSpaceNet with MobileNet as a backbone, s denotes convolution stride and k denotes kernel size.

limit of Eq. 6.5 as $\bar{\sigma}^2$ approaches zero.

$$\begin{aligned}&\lim_{\bar{\sigma}^2\to 0}\left(\int g(\bar{x})f(\bar{x}|\bar{\mu},0)d\bar{x}\right)\\&=\int g(\bar{x})\delta(\bar{x}-\bar{\mu})d\bar{x}=g(\bar{\mu})\end{aligned} \tag{6.6}$$

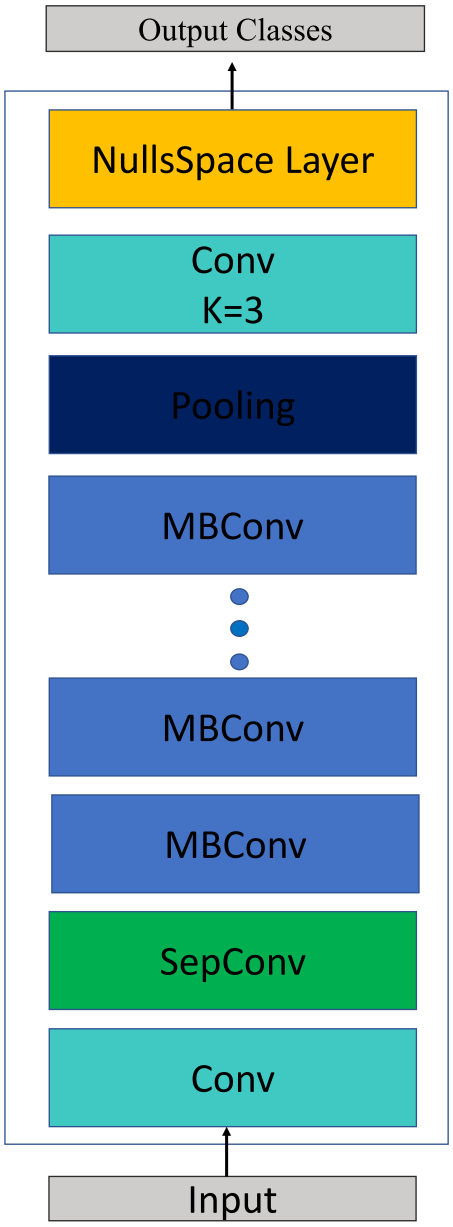


Figure 6.3: NullSpaceNet with MnasNet as a backbone. MBConv denotes mobile inverted bottleneck conv., k denots the kernel size and SepConv denotes separable conv.

where $g(\bar{\mu})$ is the dirac delta function. **Using Lemma 2 in NullSpaceNet:** Using Eqs. 6.3, 6.4, and 6.6 to find the limit of Eq. 6.1 (which guarantee the best separability as explained above), we get:

$$\lim_{(P^T S_w P) \to 0} \mathcal{J}(P) = \infty \tag{6.7}$$

Since the between-class scatter matrix S_b in Eq. 6.3 is hard to calculate, especially in the case of high dimensional features, we calculate S_b using the total-class scatter matrix S_t and the within-class scatter matrix S_w as follows:

$$S_b = S_t - S_w \tag{6.8}$$

By substituting Eq. 6.8 in Eq. 6.4 and using Eq. 6.3 we get:

$$P^\top S_t P > 0 \tag{6.9}$$

Since the output of the NullSpaceNet is $F \in \mathbb{R}^{D \times N}$ when the input batch $X \in \mathbb{R}^{W \times H \times D \times N}$, where N is the number of images. We define the within-class scatter matrix S_w and the total-class scatter matrix S_t from the output of NullSpaceNet as follows:

$$S_w = \frac{1}{N} F_w F_w^\top, \quad S_t = \frac{1}{N} F_t F_t^\top \tag{6.10}$$

where F_w is the centered class mean output features (i.e, subtracting the class mean from each feature output belonging to this class), and F_t is the centered global mean output features as shown in Eq. 6.11.

$$\begin{aligned} F_w &= (X - \mu_c) \\ F_t &= (X - \mu_g) \end{aligned} \tag{6.11}$$

Where μ_c is the class mean and μ_g is the global mean of the dataset.

Now, we want to integrate the scatter matrices we derived in Eq. 6.10 in the joint-nullspace formulation. Let U_t denote the nullspace of the total-class scatter matrix and U_w denote the nullspace of within-class matrix. From the definition of the nullspace and using the fact that S_t is non-negative definite, we get:

$$\begin{aligned} U_t &= \left\{u \in \mathbb{R}^d \mid S_t u = 0\right\} \\ &= \left\{u \in \mathbb{R}^d \mid u^\top S_t u = 0\right\} \\ &= \left\{u \in \mathbb{R}^d \mid (F_t^\top u)^\top F_t^\top u = 0\right\} \\ &= \left\{u \in \mathbb{R}^d \mid F_t^\top u = 0\right\}. \end{aligned} \tag{6.12}$$

similarly, we get U_w [2].

Lemma 3: The projection matrix P that satisfies the constraints in Eq. 6.3 and Eq. 6.4 can be achieved, if and only if, P lies in the shared space between $U_t^\perp$ and U_w, mathematically represented as:

$$P \in (U_t^\perp \cap U_w). \tag{6.13}$$

where $U_t^\perp$ is the orthogonal complement subspace of U_t spanned by the centered global mean output features. $U_t^\perp$ can be obtained using the Gram-Schmidt process [34].

Proof: Geometrically by looking at Eq. 6.12 and Eq. 1 (in the appendix), the only space that satisfies $S_t u = 0$ and $S_w u = 0$ is the joint-space where $U_t^\perp$ and U_w overlap [17].

Using Lemma 3 in NullSpaceNet:

Now we have the nullspace of S_w, which is U_w, and the nullspace of S_t, which is U_t. One problem when claculating the nullspace of S_w is that the dimensionality of the nullspace is at least $(D + C - n)$, where D is data dimensionality (which is high when we use the output of NullSpaceNet, e.g., 2048), C is the number of classes, and n is the sample size as it has been proved in [35]. To address this problem, we revert to Eq. 6.8 where it can be seen geometrically that S_t is the intersection of the nullspace of S_b and the nullspace of S_w [33]. Hence, the

[2]*See Details in Appendix Eq. 1*

nullspace of S_t can be removed based on this observation. We proceed with this solution using the Singular Value Decomposition (SVD) theory to decompose F_t as follows:

$$F_t = U\Sigma V^\top \tag{6.14}$$

Where U and V are orthogonal and U has orthonormal basis.

$$\Sigma = \begin{pmatrix} \Sigma_t & 0 \\ 0 & 0 \end{pmatrix} \tag{6.15}$$

Σ_t is the diagonal matrix $\Sigma_t \in \mathbb{R}^{t\times t}$ with the Eigenvalues. Now we can represent S_t as follows:

$$\begin{aligned} S_t &= \frac{1}{N} F_t F_t^\top \\ &= \frac{1}{N} U\Sigma V^\top V \Sigma^T U^\top \\ &= \frac{1}{N} U\Sigma\Sigma^T U^\top \\ &= \frac{1}{N} U \begin{pmatrix} \Sigma_t^2 & 0 \\ 0 & 0 \end{pmatrix} U^T \end{aligned} \tag{6.16}$$

We select a portion of basis U with dimension $U_1 \in \mathbb{R}^{m\times t}$ where $t = RANK(S_t)$, using the new subspace U_1 spanned by the new set of the basis, we project the scatter matrices as follows:

$$\begin{aligned} \tilde{S}_b &= U_1 S_b U_1^\top, \\ \tilde{S}_w &= U_1 S_w U_1^\top, \text{ and} \\ \tilde{S}_t &= U_1 S_t U_1^\top \end{aligned} \tag{6.17}$$

Where $\tilde{(.)}$ represents the reduced version of the decomposed S_b, S_w, and S_t.

From Eq. 6.14 and Eq. 6.17, we can now apply the SVD on F_t with complexity of $O\left(Dn^2\right)$

instead of $O(D^2n)$. Also, instead of calculating the nullspace of S_w, we can calculate the nullspace of $\tilde{S}_w$ as shown in Eq. 6.18. This gives the network two advantages: 1) the model does not suffer from the small sample size (SSS) problem, e.g., the model has high dimensional output features while training on small batches of images, as in [32], and 2) it is faster than solving the generalized Eigenvalue problem.

$$W = Span(\tilde{S}_w) \tag{6.18}$$

Where W is the nullspace of $\tilde{S}_w$.

Finally, the projection matrix that satisfies Eq. 6.3 and Eq. 6.4 can be calculated by:

$$P = W \times M \tag{6.19}$$

Where M is the eigenvectors of $W^T\tilde{S}_bW$ corresponding to the non-zero Eigenvalues. Consequently, maximizing the Eigenvalues of $W^T\tilde{S}_bW$, by NullSpaceNet, leads to projecting the features onto the joint-nullspace.

6.4.4 Loss Function and Its Gradient

Training the NullSpaceNet requires the loss function to be differentiable everywhere. Hence, we propose a novel differentiable loss function that maximizes the positive, or minimizes the negative, of the average non-zero Eigenvalues of the decomposed $W^T\tilde{S}_bW$. Given C classes, we define E as an Eigenvalue and $k = C - 1$ is the number of Eigenvalues. The steps to calculate the proposed differentiable loss function is shown in Alg. 4. The final loss function is shown in Eq. 6.20.

$$\mathcal{L}(\phi_E(x;\theta)) = -\frac{1}{k}\sum_{i=1}^{k} SVD_{E_i}(W^T\tilde{S}_bW) \tag{6.20}$$

```
Algorithm 4: Steps to Calculate the Proposed Loss Function
input  : The output of the last layer of $VGG16$ $F \in \mathbb{R}^{D \times N}$
output: Optimize the weights of the NullSpaceNet using the proposed differentiable loss
        function $\mathcal{L}$ based on the nullspace formulation
1: Calculate matrix $F_t$;
2: Calculate $SVD(F_t^T)$;
3: Calculate the scatter matrices $S_w, S_b, S_t$;
4: Calculate matrices $\bar{S}_t, \bar{S}_b, \bar{S_w}$ from Eq. 6.17;
5: Calculate the nullspace $W$ of $\bar{S_w}$ using Eq. 6.18;
6: Solve for the Eigenvalues of $W^T \tilde{S}_b W$ using 6.19;
7: Formulate the loss function over the average of the non-zero Eigenvalues using Eq. 6.20;
8: Use the proposed differentiable loss function in Eqs. 6.20 and its derivative as shown in
   the appendix Eq. 50 to train the network
```

Where $E \in \{E_1, \ldots, E_k\} = \{E_j | E_j < \min\{E_1, \ldots, E_{C-1}\} + \epsilon\}$.

Using the chain rule [36], the derivative of the loss function in Eq. 6.20 w.r.t the last layer H is given by $\frac{\partial \mathcal{L}(\phi_E(x;\theta))}{\partial H}$ [3].

6.4.5 Insights into NullSpaceNet

In this section, we provide a deeper look, both mathematically and geometrically, into the proposed NullSpaceNet.

Mathematical Insights: The main idea of NullSpaceNet is to learn to map the input data to another subspace (different from the traditional feature space) that satisfies the two constraints in Eq. 6.3 and Eq. 6.4. The new proposed subspace (i.e., the joint-nullspace) mathematically forces the within-class scatter matrix to vanish through the optimization of the proposed loss function in Eqs. 6.20. Meanwhile the new joint-nullspace mathematically forces the between-class scatter matrix to always be positive through the optimization of the loss function in Eq. 6.20.

Geometric Insights: The features that are produced by NullSpaceNet are living in the hyperplane represented by $U_t^{\perp} \cap U_w$, as shown in Fig. 6.4. The hyperplane is now well-defined and all the features are located in a confined space that can be precisely described both mathematically and geometrically.

Based on the above insights discussion, this proves our claim that same class inputs are collapsed into one point in the joint-nullspace, and the different classes are collapsed into different points with high separation margins.

[3] *See Details in Appendix Eq. 50*

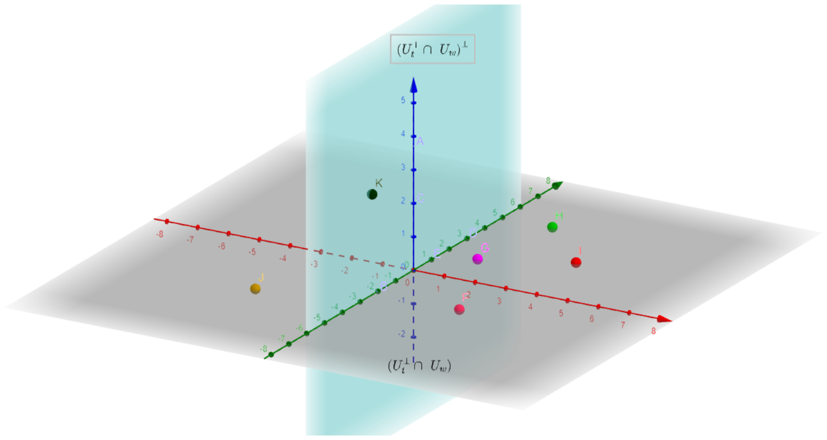


Figure 6.4: A geometric illustration of the learned features projected by the network in the joint-nullspace. Each color associated with a letter encodes a class, note that all inputs from the same class are collapsed into a single point. Notice that all classes exist on the Grey-colored hyperplane $(U_t^{\perp} \cap U_w)$ including point k.

6.4.6 Using other Feature Extractors

NullSpaceNet formulation can be applied to different feature extractor backbones. The only difference is that the last layer should be of spatial size $F \in \mathbb{R}^{D\times 1}$. For example, in case of using the feature extractor of MnasNet, a convolutional layer with kernel size=3 is used to produce a 2D tensor of shape $D \times 1$ as shown in Fig. 6.2. On the other hand, in the case of using the feature extractor of MobileNet, a convolutional layer with kernel size = 7 is used to produce a tensor of shape $D \times 1$ as shown in Fig. 6.3.

6.5 Training and Inference of NullSpaceNet

6.5.1 NullSpaceNet Training Phase

The input batch of images is fed into the input layer as shown in Fig. 6.1. The batch goes through NullSpaceNet's feature extractor layers, which include convolution, batch normalization, and pooling. Then, to the new nullspace layer, where all calculations and the new loss function in Eq. 6.20 are performed as shown in Alg. 4.

During the training, we keep track the mean of each class (i.e., $\mu_k = (\mu_1, ..., \mu_c)$) from the last layer which has the dimension $\mathbb{R}^{D\times N}$ using the moving average, where N is the number of images. Then, the eigenvectors of the decomposed $W^T \tilde{S}_b W$ are calculated after the training using the moving average for each class, which is then used in Eq. 6.19 to calculate the projection matrix P.

6.5.2 NullSpaceNet Inference Phase

In the inference phase, the output of NullSpaceNet t^T can be classified using the hyperplane equation below:

$$\text{argmax}_k\, \beta_k(t) = t^\top \mathbf{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^\top \mathbf{\Sigma}^{-1}\mu_k \tag{6.21}$$

Where β is the hyperplane and $\Sigma = P \times P^\top$.

6.6 Experimental Results

Implementation Details:

NullSpaceNet is defined in mixed-precision using the publically available NVIDIA APEX library [37]. NullSpaceNet was trained using 4 v100 Tesla GPUs with 32 GB and implemented in Python using PyTorch framework [38]. All experiments have been performed on Linux with Xeon E5 @2.20 GHz CPU and NVIDIA Titan XP GPU. All experiments are performed on networks which are trained from scratch. We set ϵ to 1 and the number of epochs for the training to 200. We used Adam optimizer [39] with a learning rate anneals geometrically at each epoch starting from 0.001, a momentum of 0.9, and a batch size of 400 images.

Datasets:

NullSpaceNet has been tested on ImageNet [10], CIFAR10 [40], CIFAR100 [40] and STL10 [41].

ImageNet is a large scale dataset for classification and detection. ImageNet classification has 1000 categories of natural images, it consists of 1.3 million images for the task of classification.

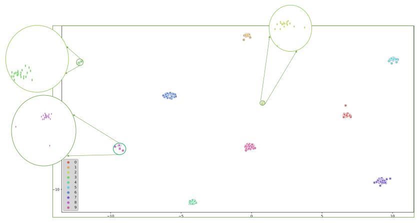
CIFAR10 and CIFAR100 have 10 and 100 classes of spatial size (32 $\times$ 32), respectively. The images were collected from natural scenes. Each dataset has 50,000 images for training and 10,000 images for testing. We used 49,000 for training, 1,000 for validation and 10,000 for testing for both datasets.

STL10 dataset has 10 classes and the resolution of images has been re-sized to (64 $\times$ 64). STL10 has 5,000 images for training, while the testing set has 8,000 images. We used 5,000 for training set, 1,000 for validation set from the testing set and the remaining 7,000 for testing.

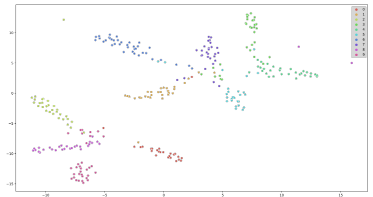
6.6.1 NullSpaceNet Results with VGG16 as backbone

Results on ImageNet:

ImageNet has a higher resolution with a spatial size of 224×224 compared to CIFAR datasets and STL10 . Results of the proposed NullSpaceNet with VGG16 as backbone compared to the



(a) NullSpaceNet



(b) VGG16+FC

Figure 6.5: t-SNE visualization of the learned features on STL10 dataset using (a) NullSpaceNet, (b) VGG16+FC.

Table 6.1: Test Accuracy on ImageNet (resolution: 224 $\times$ 224).

Architecture	Top-1 Accuracy	Top-5 Accuracy	# Params.	Avg. inf. time/batch
$VGG16$ + FC (cross-entropy loss)	74.4 %	91.0%	134,309,962	3.1556
NullSpaceNet (Proposed)	**75.7%**	**93.2%**	**18,411,936**	0.0262

traditional VGG16 with FC tested on ImageNet are shown in Table 6.1. The number of parameters have been reduced from ≈134 Million in VGG16+FC to ≈18 Million in NullSpaceNet. As it can be seen from Table 6.1, the accuracy gain is in favor of NullSpaceNet. NullSpaceNet has a Top-1 accuracy of 75.7% while VGG16+FC has a top-1 accuracy of 74.4%, which is a gain of 1.3%. The average inference time has significantly reduced with a rate of 99.17%.

Results on CIFAR10 Dataset

The results on CIFAR10 dataset are shown in Table 6.2. VGG16+FC achieves an accuracy of 93.51%, while the proposed NullSpaceNet achives 94.01%. The accuracy difference between the proposed NullSpaceNet and the VGG16+FC is $\approx 0.5\%$, in favor of NullSpaceNet.

More importantly, there is a significant reduction in the network parameters of NullSpaceNet

Table 6.2: Test Accuracy on CIFAR10.

Architecture	Accuracy	# Params.	Avg. inference time/batch
$VGG16$ + FC (cross-entropy loss)	93.51%	134,309,962	0.6841
NullSpaceNet (Proposed)	**94.01%**	**18,411,936**	**0.0051**

compared to VGG16+FC. The parameters went down from ≈ 134 Million parameters in VGG16+FC to ≈ 18 million in NullSpaceNet, which is a reduction of 86.29%. Moreover, Table 6.2 shows that the inference time required per batch by VGG16+FC is 0.6841 seconds while NullSpaceNet required only 0.0051 seconds, which is a reduction of 99.25% in favor of NullSpaceNet.

Results on CIFAR100 Dataset

The results on CIFAR100 dataset are shown in Table 6.3. NullSpaceNet outperforms VGG16+FC by gain of 0.07% in terms of accuracy (the gain is not significant similar to the CIFAR10 dataset). However, the number of parameters in NullSpaceNet has been reduced from ≈ 134 Million parameters to ≈ 18 million parameters. Moreover, Table 6.3 shows that the inference time required per batch by VGG16+FC is 0.6841 seconds while NullSpaceNet required only 0.0051 Seconds, which is a reduction of 99.25% in favor of NullSpaceNet.

The importance of conducting this experiment on CIFAR100 dataset is to prove that NullSpaceNet performance is not significantly affected by the increase in the number of classes in the classification task.

Results on STL10 Dataset

The results on STL10 dataset are shown in Table 6.4. NullSpaceNet outperforms VGG16+FC in terms of accuracy with gain of 2.57%, parameters reduction of 86.29%, and inference time reduction of 99.22%. It is worth noting that NullSpaceNet significantly benefits from the higher

Table 6.3: Test Accuracy on CIFAR100.

Architecture	Accuracy	# Parameters	Avg. inference time/batch
$VGG16$ + FC (cross-entropy loss)	92.26%	134,309,962	0.6841
NullSpaceNet(proposed)	**92.33%**	**18,411,936**	**0.0051**

Table 6.4: Test Accuracy STL10 dataset.

Architecture	Accuracy	# Parameters	Avg. inference time/batch
$VGG16$ + FC (cross-entropy loss)	93.74%	134,309,962	1.3487
NullSpaceNet (proposed)	**96.31%**	**18,411,936**	**0.0105**

image resolution, STL10 has an image resolution of 64 $\times$ 64.

Visualization

To visualize the learned features by NullSpaceNet and VGG16+FC on STL10 dataset, t-SNE is used to produce Fig. 6.5. Each color is associated with a number that represents a class in the STL10 dataset. It can be seen from Fig. 6.5 (a) that the within-class scatter for all classes has been reduced to the minimum and the between-class scatter has been maximized with high margin separation among all classes. Fig. 6.5 (a) visualizes the power of the learned joint-nullspace that has the optimal separation among different classes. By examining Fig. 6.5 (b), the classes are overlapping and the separation margin is not optimal.

6.6.2 NullSpaceNet with other Feature Extractor Backbones

Another two experiments have been conducted on MobileNet [42] and MnasNet [43] as the backbones of NullSpaceNet. The architectures of NullspaceNet with MobileNet and MnasNet as backbones are shown in Fig. 6.2 and Fig. 6.3, respectively.

Table 6.5 shows that the MobileNet network has 5.4 Million parameters with an accuracy of 70.6%. In the case of NullSpaceNet with MobileNet as the backbone, the accuracy went up to 72.30% and a reduction in the number of parameters of $\approx 8.7\%$ has been achieved. Consequently, the inference time dropped down by 70.67%. It is clear that NullSpaceNet with the MobileNet feature extractor backbone does not significantly reduce the number of parameters due to the last fully connected layer in the original MobileNet, which only has 469,460 parameters. However, the inference time has been significantly reduced.

Similarly, Table 6.5 shows that NullSpaceNet with MnasNet backbone has a gain of 1.7% in terms of accuracy and a number of parameters reduction rate of 27.95%. The average inference time of NullSpaceNet with MnasNet as a backbone has been significantly reduced by 69.05%. This confirms that NullSpaceNet can be applied to different backbones with the benefit of parameter reductions and inference time.

6.7 Results and Discussion

Impact of Image Resolution

The accuracy gain between the proposed NullSpaceNet with VGG16 as backbone and the traditional VGG16+FC when tested on CIFAR10 and CIFAR100 is 0.5% and 0.07%, respectively, in favor of NullSpaceNet. The gain suggests that the accuracy does not significantly benefit

Table 6.5: Test Accuracy on ImageNet dataset using different backbones with image resolution 224×224.

Architecture	Top-1 Accuracy	Top-5 Accuracy	# Params.	Avg. Inf. time/batch
MobileNet-224	70.60%	89.50%	5,400,000	0.300 Seconds
NullSpaceNet(MobileNet-224)	**72.30%**	**92.41%**	**4,930,540**	**0.088 Seconds**
MNASNET1-0	75.20%	92.50%	3,900,000	0.210 Seconds
NullSpaceNet(MNASNET1-0)	**76.80%**	**93.70%**	**2,810,140**	**0.065 Seconds**

Table 6.6: A summary of comparing NullSpaceNet vs. VGG16+FC on 5 datasets, CIFAR10, CIFAR100, STL10 and its modified version, and ImageNet. All results are in favor of NullSpaceNet

Dataset	Accuracy Difference	Image Size	Params. Reduction	Time Reduction
CIFAR10	+0.50%	32×32	+86.29%	+99.25%
CIFAR100	+0.07%	32×32	+86.29%	+99.25%
STL10 (64x64)	+2.57%	64×64	+86.29%	+99.22%
STL10 (modified 32x32)	+0.02%	32×32	+86.29%	+99.25%
ImageNet	+1.30%	224×224	+86.29%	+99.17%

from the projection onto the proposed joint-nullspace in this case. This can be justified based on the fact that the image resolution in CIFAR10 and CIFAR100 is 32×32. This means that the number of pixel-level features to be mapped to either the feature space or the joint-space is

Table 6.7: Test Accuracy on modified version of STL10 (resolution: 32 $\times$ 32).

Architecture	Accuracy	$\#$ Params.	Avg. inference time
$VGG16$ + FC (cross-entropy loss)	93.89$\%$	134,309,962	0.6841
NullSpaceNet (Proposed)	**93.91$\%$**	**18,411,936**	**0.0051**

small, and hence explains the low accuracy gain.

This justification is further supported in light of the results on the STL10 (which has higher image resolution of 64×64), and thus better accuracy in favor of NullSpaceNet.

Furthermore, another experiment has been performed on a reduced resolution version of STL10. All training images have been reduced to 32 $\times$ 32 resolution, similar to CIFAR10 and CIFAR100. NullSpaceNet has been trained on the modified version of STL10, and the results are shown in Table 6.7. It is seen that the accuracy gain drops to 0.02%, similar to the ones in CIFAR10 and CIFAR100.

Impact of Image Resolution

It is clear from Table 6.1 that the NullSpaceNet has a gain of $+1.3$ in terms of Top-1 accuracy compared to VGG16+FC on ImageNet. ImageNet has 1000 classes and the spatial size of images is 224×224. This confirms our justification that NullSpaceNet has better accuracy in cases of images with higher resolution. In general, NullSpaceNet outperforms VGG16+FC in all cases. All results are summarized in table 6.6

Top-k Error Rate on STL10

Top-k error rate is the fraction of the testing set for which the true label is not among the five labels that are most likely by the model prediction [10]. Fig. 6.6 shows top-1 accuracy on STL10 dataset. It is clear from Fig. 6.6 that NullSpaceNet has a lower error rate compared

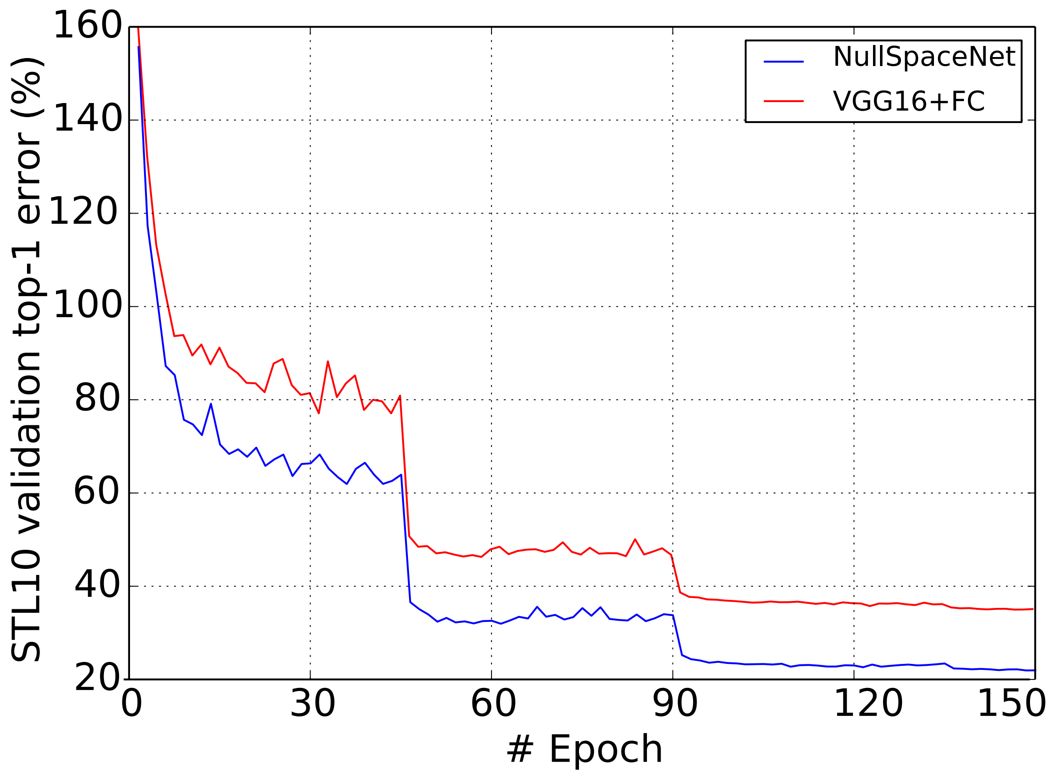


Figure 6.6: Top-1 error on the validation set of STL10. At the beginning, NullSpaceNet and VGG16+FC have almost the same error rate, then NullSpaceNet has a lower error rate as the number of epochs increase.

to VGG16+FC. In Fig. 6.7, the top row shows the training and testing loss over three datasets CIFAR10, CIFAR100, and STL10. It is clear that NullSpaceNet converges to the minimum at 200 epochs without overfitting to the training set. In the bottom row, we report the top-1 and top-5 accuracy on CIFAR10, CIFAR100, and STL10. The Top-1 and Top-5 accuracy show that NullSpaceNet is robust to image challenges as the top-5 accuracy is always high.

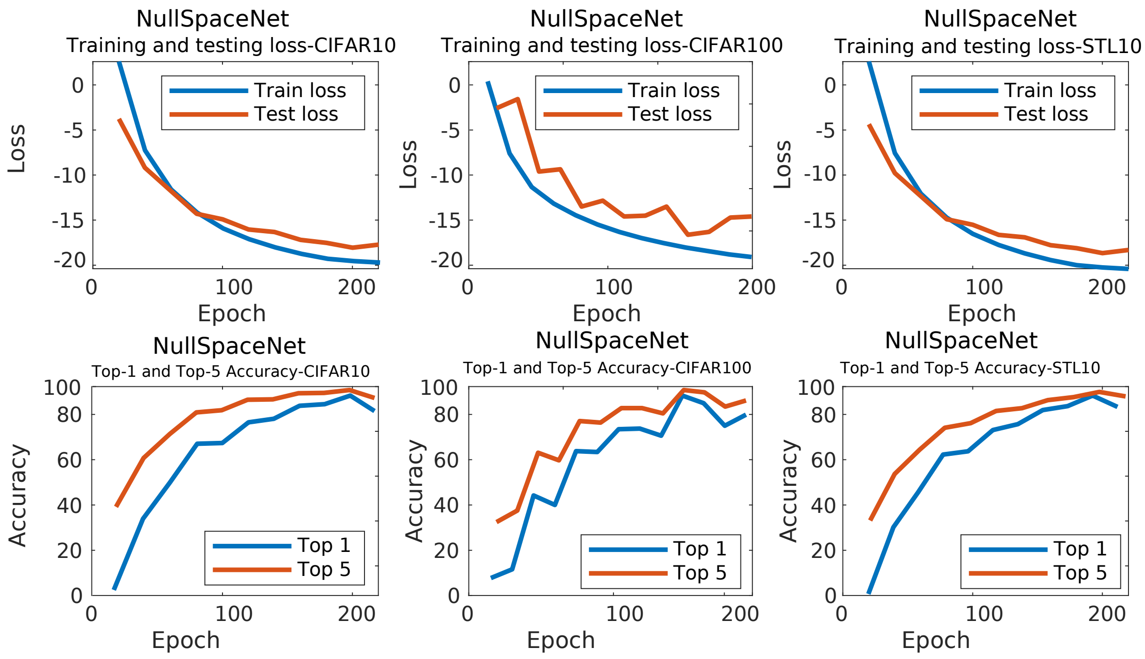


Figure 6.7: Accuracy and Loss on CIFAR10, CIFAR100, and STL10. Top row: Training and testing losses. Bottom row: the top-1 and top-5 accuracy.

6.8 Conclusions and Future Work

A typical CNN optimizes the weights of the network by maximizing the likelihood between the estimated probability of the predicted class and the true probability of the correct class. NullSpaceNet learns to project the features from the pixel-level (i.e, input image) to a joint-nullspace. All features from the same class are collapsed into a single point in the learned joint-nullspace, whereas all features from different classes are collapsed into different points with high separation margins. Also, a novel differentiable loss function is developed to train NullSpaceNet to learn to project the features onto the joint-nullspace. NullSpaceNet with the proposed differentiable loss function exhibits superior performance, with accuracy gains of

$0.02 - 2.57\%$, and a reduction in inference time of $\approx 99 - 70\%$ in favor of NullSpaceNet. This means that NullSpaceNet needs $1 - 30\%$ of the time it takes a traditional CNN to classify a batch of images with competitive accuracy. Future work includes extending this work to other fields such as object tracking. The future work will formulate the joint-nullspace as a learnable regularizer and auxiliary loss to merge the feature space and nullspace in one network. Formulating the regularizer in this way, will leverage the best of both spaces.

References

[1] M. H. Abdelpakey and M. S. Shehata, "Domainsiam: Domain-aware siamese network for visual object tracking," in *International Symposium on Visual Computing*. Springer, 2019, pp. 45–58.

[2] M. H. Abdelpakey and M. S. Shehata, "Dp-siam: Dynamic policy siamese network for robust object tracking," *IEEE Transactions on Image Processing*, vol. 29, pp. 1479–1492, 2019.

[3] M. M. Mohamed, "Denssiam: End-to-end densely-siamese network with self-attention model for object tracking," in *Advances in Visual Computing: 13th International Symposium, ISVC 2018, Las Vegas, NV, USA, November 19–21, 2018, Proceedings*, vol. 11241. Springer, 2018, p. 463.

[4] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6182–6191.

[5] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time

6-dof camera relocalization," in *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2938–2946.

[6] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[7] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4207–4215.

[8] H. Huang, C. Wang, P. S. Yu, and C.-D. Wang, "Generative dual adversarial network for generalized zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 801–810.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] Y. Zheng, D. Liu, Q. Ren, B. Sun, and Z. Niu, "Object tracking via null-space discriminative projections and sparse representation," in *Proceedings of the International Conference on Video and Image Processing*, 2017, pp. 243–248.

[12] L. V. Foster, “Rank and null space calculations using matrix decomposition without column interchanges,” *Linear Algebra and its Applications*, vol. 74, pp. 47–71, 1986.

[13] B. Recht, W. Xu, and B. Hassibi, “Null space conditions and thresholds for rank minimization,” *Mathematical programming*, vol. 127, no. 1, pp. 175–202, 2011.

[14] R. Kueng and P. Jung, “Robust nonnegative sparse recovery and the nullspace property of 0/1 measurements,” *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 689–703, 2017.

[15] A. ElTantawy and M. S. Shehata, “Local null space pursuit for real-time moving object detection in aerial surveillance,” *Signal, Image and Video Processing*, vol. 14, no. 1, pp. 87–95, 2020.

[16] F. Dufrenois and J.-C. Noyer, “A null space based one class kernel fisher discriminant,” in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 3203–3210.

[17] Y.-F. Guo, L. Wu, H. Lu, Z. Feng, and X. Xue, “Null foley–sammon transform,” *Pattern recognition*, vol. 39, no. 11, pp. 2248–2251, 2006.

[18] J. Wen, X. Fang, J. Cui, L. Fei, K. Yan, Y. Chen, and Y. Xu, “Robust sparse linear discriminant analysis,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 390–403, 2018.

[19] P. Hu, D. Peng, Y. Sang, and Y. Xiang, “Multi-view linear discriminant analysis network,” *IEEE Transactions on Image Processing*, vol. 28, no. 11, pp. 5352–5365, 2019.

[20] Q. Hou, Y. Wang, L. Jing, and H. Chen, “Linear discriminant analysis based on kernel-based possibilistic c-means for hyperspectral images,” *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 8, pp. 1259–1263, 2019.

[21] Z. Fei, E. Yang, D. D.-U. Li, S. Butler, W. Ijomah, X. Li, and H. Zhou, “Deep convolution network based emotion analysis towards mental health care,” *Neurocomputing*, 2020.

[22] Q. Wang, Z. Meng, and X. Li, “Locality adaptive discriminant analysis for spectral–spatial classification of hyperspectral images,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 11, pp. 2077–2081, 2017.

[23] S. A. Korkmaz, H. Bínol, A. Akçiçek, and M. F. Korkmaz, “A expert system for stomach cancer images with artificial neural network by using hog features and linear discriminant analysis: Hog_lda_ann,” in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*. IEEE, 2017, pp. 000 327–000 332.

[24] M. Mahdianpari, B. Salehi, F. Mohammadimanesh, B. Brisco, S. Mahdavi, M. Amani, and J. E. Granger, “Fisher linear discriminant analysis of coherency matrix for wetland classification using polsar imagery,” *Remote Sensing of Environment*, vol. 206, pp. 300–317, 2018.

[25] Y. Liang, L. Sun, W. Ser, F. Lin, S. T. G. Thng, Q. Chen, and Z. Lin, “Classification of non-tumorous skin pigmentation disorders using voting based probabilistic linear discriminant analysis,” *Computers in biology and medicine*, vol. 99, pp. 123–132, 2018.

[26] A. ElTantawy and M. S. Shehata, “Krmaro: aerial detection of small-size ground moving objects using kinematic regularization and matrix rank optimization,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 6, pp. 1672–1686, 2018.

[27] J. Liu, Z. Lian, Y. Wang, and J. Xiao, "Incremental kernel null space discriminant analysis for novelty detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 792–800.

[28] T. Pang, C. Du, and J. Zhu, "Max-mahalanobis linear discriminant analysis networks," *ICML*, 2018.

[29] L. Zhang, M. Edraki, and G.-J. Qi, "Cappronet: Deep feature learning via orthogonal projections onto capsule subspaces," in *Advances in Neural Information Processing Systems*, 2018, pp. 5814–5823.

[30] H. Pan and H. Jiang, "Learning convolutional neural networks using hybrid orthogonal projection and estimation," in *Proceedings of the Ninth Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M.-L. Zhang and Y.-K. Noh, Eds., vol. 77. PMLR, 15–17 Nov 2017, pp. 1–16. [Online]. Available: http://proceedings.mlr.press/v77/pan17a.html

[31] Q. Tian, T. Arbel, and J. J. Clark, "Deep lda-pruned nets for efficient facial gender classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 10–19.

[32] M. Dorfer, R. Kelz, and G. Widmer, "Deep linear discriminant analysis," *in Proceedings of the International Conference on Learning Representations (ICML)*, pp. 1–13, 2016.

[33] D. H. Foley and J. W. Sammon, "An optimal set of discriminant vectors," *IEEE Transactions on computers*, vol. 100, no. 3, pp. 281–289, 1975.

[34] V. G. Jensen, H. Jenson, and G. Jeffreys, “Mathematical methods in chemical engineering,” 1977.

[35] L.-F. Chen, H.-Y. M. Liao, M.-T. Ko, J.-C. Lin, and G.-J. Yu, “A new lda-based face recognition system which can solve the small sample size problem,” *Pattern recognition*, vol. 33, no. 10, pp. 1713–1726, 2000.

[36] E. W. Swokowski, *Calculus with analytic geometry*. Taylor & Francis, 1979.

[37] Nvidia, “Nvidia/apex,” July 2020. [Online]. Available: https://github.com/NVIDIA/apex

[38] A. Paszke, S. Gross, S. Chintala, and G. Chanan, “Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration,” *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, vol. 6, 2017.

[39] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[40] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.

[41] A. Coates, A. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[42] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.

[43] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “Mnasnet: Platform-aware neural architecture search for mobile,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

Chapter 7

NullSpaceRDAR: Regularized Discriminative Adaptive NullSpace for Robust Visual Object Tracking

7.1 Abstract

Visual object tracking is a fundamental task in many computer vision applications. Recently, discriminative-based and Siamese-based trackers have achieved outstanding performance on most recent benchmarks. However, these trackers use the pre-trained backbone networks that have been solely trained on classification without taking into consideration object tracking. These pre-trained backbones do not have a strong discriminative ability to classify the object of interest from distractors. To bridge this gap, in this paper, we propose to learn a tracker, dubbed NullSpaceRDAR, with a regularized discriminative joint-nullspace backbone network that is specifically designed for object tracking. In the regularized discriminative joint-nullspace, the

features from the same target are collapsed into one point in the joint-nullspace and different target-specific features are collapsed into different points in the joint-nullspace. Consequently, the joint-nullspace forces the network to be sensitive to the variations of the object from the same class (intra-class variations). Moreover, a dynamic adaptive loss is proposed to select the suitable loss function from a super-set family of losses based on the training data to make NullSpaceRDAR more robust to different challenges. Extensive experiments have been conducted on five benchmarks to evaluate NullSpaceRDAR: OTB100, VOT variations (i.e., VOT2015, VOT2016, VOT2018, and VOT2019), LaSOT, TrackingNet, and GOT10k. The overall results show that NullSpaceRDAR outperforms the current state-of-the-art trackers

7.2 Introduction

Visual Object tracking is the task of finding the trajectory of an arbitrary target over time. Visual object tracking has gained much attention in the last few years due to it is being a fundamental task in many computer vision applications such as autonomous driving [1], point clouds [2], adversarial attacks [3], surveillance [4].

Recently, many trackers have achieved excellent performance with examples of these including SiamRPN [5], SiamMask [6], SiamRPN++ [7], and DP-Siam [8]. However, these trackers use backbone networks that have been trained on ImageNet [30] for classification without taking into consideration the object tracking task. Moreover, these trackers do not fully utilize the background information, which makes the tracker more robust to distractors. Consequently, there is a trade-off between the three performance evaluation metrics: 1) the accuracy (A), 2) the expected average overlap (EAO), and 3) the robustness (R) (see Section 7.7 for more details on these metrics). For example, according to Table 7.5, ATOM [12] is ranked as the top tracker in terms of Accuracy (0.603), however its robustness (0.411) is ranked sixth out of 9 trackers in the table. Similarly, SiamRPN++ [7] is ranked as the second-best tracker in Table 7.5 in terms of accuracy (0.599), however its Expected Average Overlap (EAO) and Robustness (R) are ranked seventh and eighth.

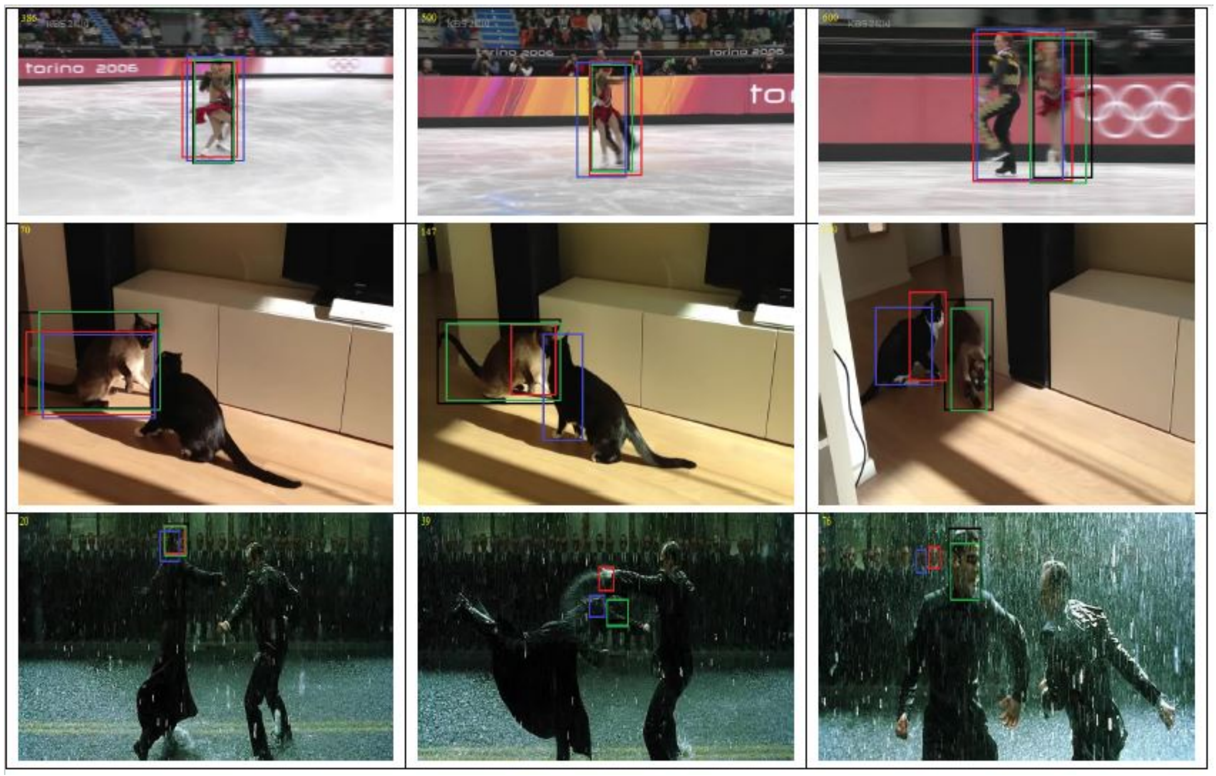
To achieve a superior performance in all three metrics (i.e., A, EAO and R), this paper argues that the best tracker should have a strong discriminative ability by using background information to classify the target from the background and accurately estimate the target location. To achieve this strong discriminative power, this paper proposes to transfer the backbone features from the traditional feature space to a novel joint-nullspace with strong discriminative power ability.

This paper introduces NullSpaceRDAR, a tracker that learns the backbone's features in a joint-nullspace instead of feature space. The joint-nullspace will ensure that features from the same target-specific are collapsed into one point in the joint-nullspace. Moreover, features from different target-specific information are collapsed into different points in the joint-nullspace. Hence, this formulation ensures a high discriminative power that effectively enables the tracker to separate the target from the background with high discriminative ability.

To summarize, the two main contributions of this paper are:

- First, a novel formulation is proposed for the feature learning in the backbone network by projecting the features onto a joint-nullspace. This new formulation enables a high discriminative ability due to the large separation margin among the different points in the learned space, while grouping the same target-specific information points with extremely-low separation margins in the learned space.

- Second, a novel dynamic loss is proposed to adaptively switch between different loss functions based on the training data.



- Ground truth - NullSpaceRDAR (proposed) - DiMP - ATOM

Figure 7.1: Results of the proposed tracker (NullSpaceRDAR), ATOM, and DiMP on three different sequences from VOT2019. The columns contain the output of the trackers over time, while the rows contain iceekater2, Fernando, and matrix sequences

The source code and results are made publicly available[1]. The rest of the paper is organized as follows. Section 7.3 introduces the related work. Section 7.4 details the proposed approach. The experimental results are presented in Section 7.7. Finally, Section 7.8 concludes the paper and presents future directions.

[1] https://vip-mun.github.io/NullSpaceRDAR/

7.3 Related work

In recent years, deep learning has revolutionized visual object tracking and advanced state-of-the-art performance. For example, Siamese-based trackers have achieved outstanding performance with high speed. Also, there is another family of discriminative-based trackers that incorporate discriminative classifiers in their formulation and also achieved outstanding performance.

Siamese-based Object Tracking:

The seminal work SiamFC in this category was proposed in [13]. SiamFC formulates the object tracking problem as a similarity metric learning for prediction. The network consists of two branches, namely the target branch and the search branch. The target branch takes in the target image to produce a filter-like for the feature map of the search branch. The search branch takes in the test image and produces a feature map that is generally has a larger spatial size compared to the target branch. The filter-like and the feature map of the search branch are combined using a cross-correlation layer which produces a score map to locate the object of interest. The higher the score, the more likely the object of interest is located within the indicated area. CFNet [14] used a correlation filter in SiamFC in an end-to-end fashion to train the whole network for object tracking. The correlation filter in CFNet is differentiable and has a closed-form solution. DensSiam [15] uses the Dense-block [16] to transform the backbone network to a Densely-Siamese network. Moreover, DensSiam uses a self-attention mechanism in the target branch to boost the appearance model. SiamRPN [5] was the first work to use RPN [17] on top of Siamese-based trackers. SiamRPN uses RPN with classification and regression branches instead of relying on multi-scale search which is time consuming. SiamRPN works beyond realtime at 160 frames per second (FPS) due to its local one-shot detection formulation.

Due to the limitations of Siamese-based trackers which take advantage of deeper networks such as ResNET50 [18], SiamRPN++ [7] proposed to use a spatial-aware sampling strategy to overcome the problem of lacking strict translation invariance. SiamRPN++ is a follow-up work on SiamRPN that has achieved outstanding performance compared to SiamRPN. SiamRCNN [19] uses the re-detection framework in object tracking, the two-stage object detection approach is combined using tracklet-based dynamic programming. SiamRCNN uses the appearance model in the first frame template and the previous successful frame to model the historical appearance model. SiamBAN [20] takes advantage of the fully convolutional network to model the object tacking as a parallel classification and regression problem in one network. SiamBAN does not assume a prior bounding boxes to avoid the associated hyper-parameters. SiamCAR [21] uses a per-pixel manner and decomposes the visual tracking problem into a classification-based pixel and regression-based pixel. With these settings, SiamCAR uses two networks: Siamese network for feature extraction and a network for classification and regression to avoid hyper-parameter tuning. SiamCAR does not use RPN or anchors for bounding boxes.

Despite the fact that Siamese-based trackers achieve outstanding speed and a good accuracy, the background information is not utilized.

Discriminative-based Object Tracking:

In this category, we include trackers that use online learning and discriminative-based learning, including correlation filter trackers. Meta-learning can be used in discriminative-based trackers in offline training to learn how to update the model, similar to [22–27], and [28] . ATOM [12] divides the object tracking problem into two sub-tasks, object classification and target estimation. In object classification, the objective is to separate the target from the background by training a discriminative classifier online. In target estimation, the objective is to precisely locate the object of interest by using overlap maximization optimization in offline training. The

DiMP [10] tracker is built based on ATOM, where the network is derived from a discriminative learning loss within an optimization problem. DiMP is designed to predict the parameters of the model in an online manner with a few iterations. PrDim [29] is a follow-up work to improve the performance of DiMP tracker. In PrDimp tracker, a probabilistic regression framework is adapted in the target estimation. Thus, rather than estimating the target location confidence, PrDiMP estimates the conditional probability density of target's location.

Even though these trackers incorporate background information, their performance is still prone to inaccuracies and failure due to the lack of strong discriminative abilities.

7.4 Proposed Approach

NullSpaceRDAR consists of two phases: 1) Feature projection onto a joint-nullspace, and 2) Adaptive bounding box estimation, as shown in Fig. 7.2. In the feature projection onto a joint-nullspace phase, the backbone is first trained offline on ImageNet [30] to map from the feature space to the proposed highly discriminative joint-nullspace by regularizing the categorical cross-entropy with a differentiable regularizer. The regularizer encourages the network to minimize the within-target-specific scatter matrix to be zero and the between-target-specific scatter matrix to be positive and large.

In the adaptive bounding box estimation phase, the network learns to classify the target (using the online classifier) and accurately estimates its location (using the target estimator).

7.4.1 Feature Projection onto Joint-Nullspace Phase

In this phase, NullSpace backbone is trained offline on classification using the ImageNet [30] dataset to learn a strong discriminative joint-nullspace instead of the traditional feature space.

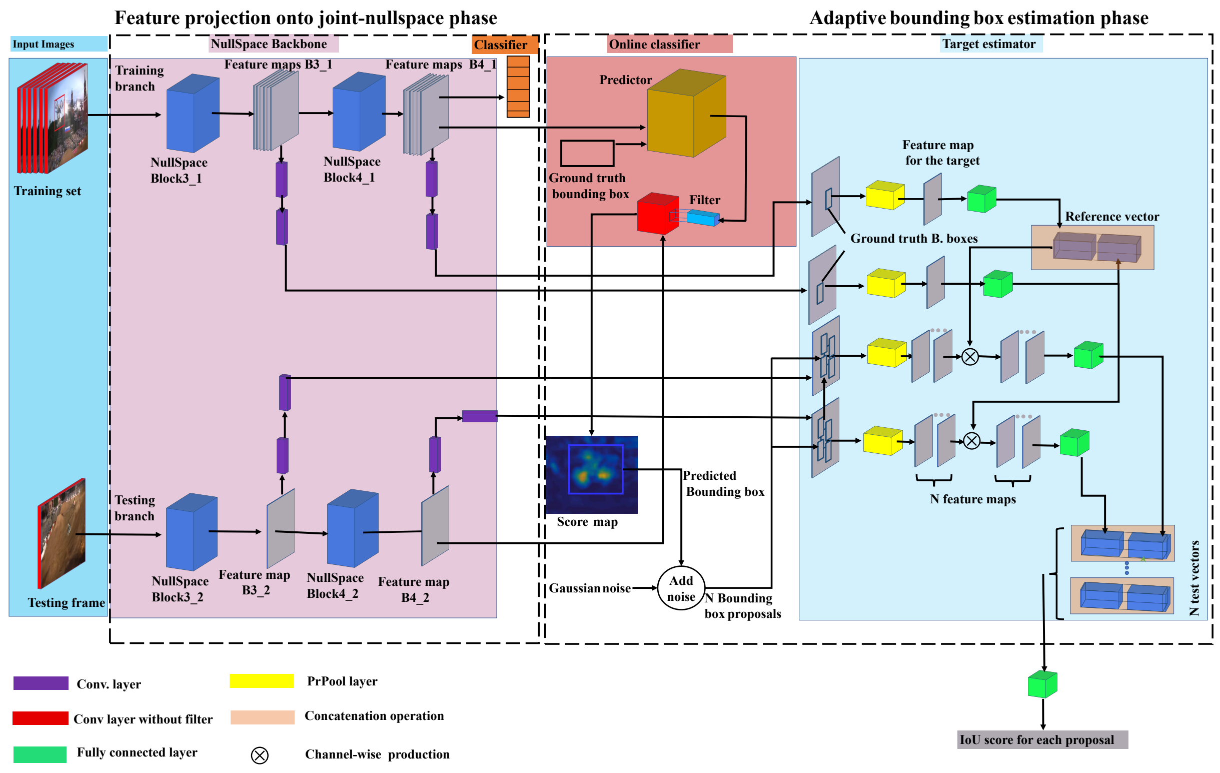


Figure 7.2: NullSpaceRDAR architecture

Fig. 7.4 shows an example of projecting 10 classes onto the traditional feature space and the joint-NullSpace. Only 10 classes were projected for the sake of visual clarity.

7.4.1.1 Mathematical Formulation

The training set (i.e., the input batch) is fed into the backbone network as shown in Fig. 7.2 and passes through the NullSpace blocks to output category-confidence from the classifier. Let the training set $X_{train} = \{x_1, x_2, ..., x_i, ..., x_N\}$ and $x_i \in \mathbb{R}^{w \times h \times d}$, where w, h, d are the $width$, $hight$ and the $depth$ of the input image, respectively. Let the loss function be defined as the

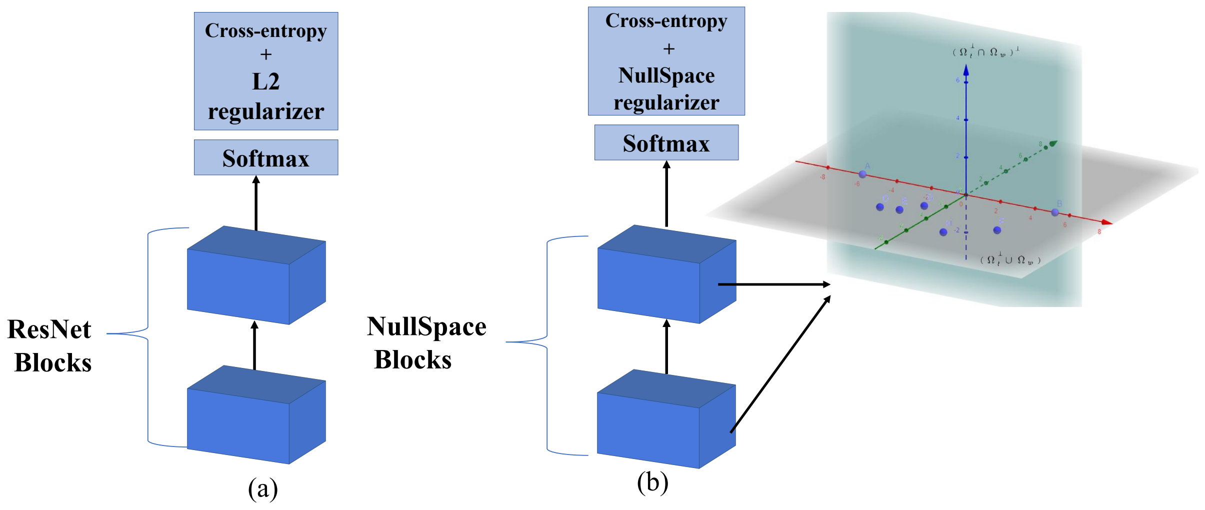


Figure 7.3: Diagram of ResNet blocks (backbone) and NullSpace blocks (backbone) (a) ResNet diagram, the loss function is regularized by the mean L^2 regularizer of the network's weights while (b) Diagram of NullSpace blocks, the loss function is regularized by nullspace of the network's weight.

categorical cross-entropy loss given by:

$$\mathcal{L}(f(\boldsymbol{x_i};\boldsymbol{\Theta}),y_{\boldsymbol{i}}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c}\boldsymbol{y}_{ij}\log f_j\left(\boldsymbol{x}_i;\boldsymbol{\Theta}\right) \tag{7.1}$$

Where f is the backbone network, x_i is the input image, $\boldsymbol{\Theta}$ corresponds to the parameters of the backbone network, y_i is the ground truth label of image x_i, and y_{ij} is the j^{th} element of the one-hot label vector corresponds to x_i. Let the regularized discriminative loss function be:

$$\mathcal{L}_r(f(\boldsymbol{x_i};\boldsymbol{\Theta}),y_{\boldsymbol{i}}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c}\boldsymbol{y}_{ij}\log f_j\left(\boldsymbol{x}_i;\boldsymbol{\Theta}\right) + \lambda\boldsymbol{R}(\boldsymbol{\Theta}) \tag{7.2}$$

Where λ is a scalar value that corresponds to the regularization strength and $\boldsymbol{R}(\boldsymbol{\Theta})$ is the regularizer function. Note that the first component of the regularized loss function (i.e., the categorical cross-entropy) uses the output of the classifier along with the labels to calculate the

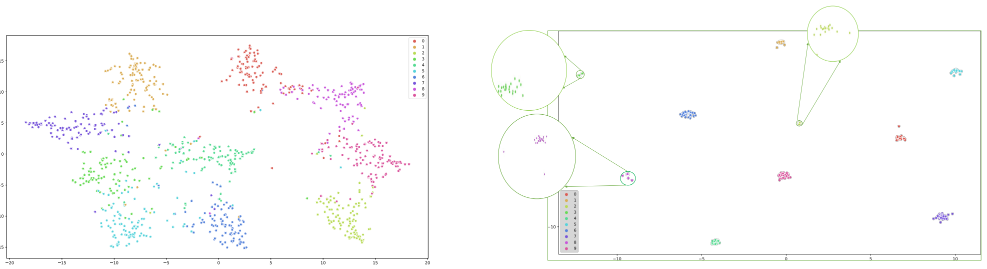


Figure 7.4: The projection from feature space to joint-nullspace. Left: the feature space of the backbone network before projection. Right: the projection onto joint-nullsapce. For the sake of visual clarity, only ten classes have been projected.

cross-entropy loss. The objective is to find a differentiable regularizer function that is robustly discriminative and encourages similar target-specific information to collapse into one point. Moreover, different target-specific information is collapsed into different points with high separation margin.

7.4.1.2 Regularized Discriminative Joint-Nullspace

In this setting, we use the output feature vector of the last block ($NullSpaceBlock4\_1$), which has a size of $F \in \mathbb{R}^{D\times1\times1}$, where $D = 2048$. In the case of a batch of images, the spatial size of $NullSpaceBlock4\_1$ becomes $F \in \mathbb{R}^{D\times N}$ for a batch of size N. To find a regularizer function that minimizes the within-target-specific scatter matrix and maximizes the between-target-specific scatter matrix, Linear Discriminant Analysis (LDA) [31] is used as a starting point for derivation. In the context of visual object tracking, the objective is seeking a learnable projection matrix B that minimizes the within-target-specific scatter matrix for similar objects while maximizing the between-target-specific scatter matrix as follows:

$$\mathcal{J}(B) = \frac{B^\top t_b B}{B^\top t_w B} \tag{7.3}$$

Where $\mathcal{J}$ is the objective function and t_b and t_w are the between-target-specific scatter matrix and the within-target-specific scatter matrix, respectively.
The closed-form solution of Eq. 7.3 is given by,

$$t_b B = \mathcal{E} t_w B \tag{7.4}$$

Eq. 7.4 is the solution for the generalized eigenvalue, where $\mathcal{E}$ is the eigenvalue of the eigenvector ${t_w}^{-1} t_b$. To encourage the network to minimize the within-target-specific scatter matrix and maximize the between-target-specific scatter matrix, we restrict the nominator and denominator of Eq. 7.3 as follow,

$$\begin{aligned} &B^\top t_b B > 0 \\ &B^\top t_w B = 0 \end{aligned} \tag{7.5}$$

These restrictions encourage the same target-specific information to collapse into one point and the different target-specific information to collapse into different points with high separation margins. By taking the limit of Eq. 7.3 and substituting by Eq. 7.5 bottom equation, we get:

$$\lim_{(B^T t_w B) \to 0} \mathcal{J}(B) = \infty \tag{7.6}$$

Let t_t denote the total scatter matrix, where $t_t = t_b + t_w$, consequently t_b can be expressed as follows,

$$t_b = t_t - t_w \tag{7.7}$$

The relationship between the total scatter matrix and the projection matrix can be derived by substituting Eq. 7.7 in Eq. 7.5 as follows,

$$\begin{aligned} &B^T (t_t - t_w) B > 0 \\ &B^T t_t B > 0 \end{aligned} \tag{7.8}$$

The within-target-specific scatter matrix and the between-target-specific scatter matrix are calculated as follows,

$$\begin{aligned} t_w &= \tfrac{1}{N} F_w F_w^\top \\ t_t &= \tfrac{1}{N} F_t F_t^\top \end{aligned} \tag{7.9}$$

where $F_w = (F - \mu_c)$, $F_t = (F - \mu_g)$, μ_c and μ_g are the class mean and global mean over the training dataset, respectively.

Let Ω_t denote the between-target-specific scatter matrix and Ω_w denote the nullspace of the within-target-specific scatter matrix, both are defined as follow,

$$\begin{aligned} \Omega_t &= \left\{\Omega \in \mathbb{R}^d \mid t_t \Omega = 0\right\} \\ &= \left\{\Omega \in \mathbb{R}^d \mid \Omega^\top t_t \Omega = 0\right\} \\ &= \left\{\Omega \in \mathbb{R}^d \mid \left(F_t^\top \Omega\right)^\top F_t^\top \Omega = 0\right\} \\ &= \left\{\Omega \in \mathbb{R}^d \mid F_t^\top \Omega = 0\right\} \end{aligned} \tag{7.10}$$

$$\begin{aligned} \Omega_w &= \left\{\Omega \in \mathbb{R}^d \mid t_w \Omega = 0\right\} \\ &= \left\{\Omega \in \mathbb{R}^d \mid \Omega^\top t_w \Omega = 0\right\} \\ &= \left\{\Omega \in \mathbb{R}^d \mid \left(F_w^\top \Omega\right)^\top F_w^\top \Omega = 0\right\} \\ &= \left\{\Omega \in \mathbb{R}^d \mid F_w^\top \Omega = 0\right\} \end{aligned} \tag{7.11}$$

From Eqs. 7.5, 7.10, and 7.11, it is geometrically clear that the only space that satisfies these equations is a joint-nullspace. In particular, the constraints on the projection matrix (described in Eq.7.5) are satisfied, if and only if, B is located at the shared space that is composed of the orthogonal complement subspace $\Omega_t^\perp$ and Ω_w as shown in Fig. 7.3. The subspace $\Omega_t^\perp$ is produced by applying Gram-Schmidt process [32] to Ω_t. Due to the high dimensionality of the output of the network, we factorize the centred output around the global mean using Singular

Value Decomposition (SVD) as follows,

$$F_w = U_w \Sigma V^\top \tag{7.12}$$

Where U and V have orthonormal basis, from Eq.7.12, the total scatter matrix can be re-written as follows,

$$t_w = \frac{1}{N} U_w \begin{pmatrix} \Sigma_w^2 & 0 \\ 0 & 0 \end{pmatrix} U_w^\top \tag{7.13}$$

Similarly, t_t is calculated using,

$$t_t = \frac{1}{N} U_t \begin{pmatrix} \Sigma_t^2 & 0 \\ 0 & 0 \end{pmatrix} U_t^\top \tag{7.14}$$

Where Σ_w and Σ_t correspond to a diagonal matrix with eigenvalues. With this formulation, we select subspace $u_t \subset U_t$ based on the rank of t_t. Consequently, the nullspace Ω_w of t_w can be represented by a subset of orthogonal basis $\tilde{\Omega}_w$ instead of the whole space Ω_w,

$$\tilde{\Omega}_w = \text{SPAN}\left(\tilde{t}_w\right) \tag{7.15}$$

Where $\tilde{t}_w$ is the projection onto subspace u_t given by $\tilde{t}_w = u_t t_w u_t^\top$. With this formulation, the projection matrix that minimizes the within-target-specific scatter matrix and maximizes the between-target-specific scatter matrix is given by,

$$B = Eigen_{val}(\tilde{\Omega}_w u_t t_b u_t^\top \tilde{\Omega}_w^\top) \tag{7.16}$$

Where $Eigen_{val}(.)$ is the non-zero eigenvalues. Hence, the regularizer that projects the feature space onto the joint-nullspace is given by,

$$R(\boldsymbol{w}) = -\frac{1}{C-1} \sum_{i=1}^{C-1} Eigen_{val}(\tilde{\Omega}_w u_t t_b u_t^\top \tilde{\Omega}_w^\top)^2 \tag{7.17}$$

Where C is the number of classes, as an example, C=1000 classes for ImageNet. Negative sign is to maximize the values during the optimization. This is a convex regularizer that satisfies Eq. 7.5. Finally, the regularized loss function is given by,

$$\begin{aligned}\mathcal{L}_r(f(\boldsymbol{x};\boldsymbol{\Theta}),y_{\boldsymbol{i}}) = &-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c}\boldsymbol{y}_{ij}\log f_j\left(\boldsymbol{x}_i;\boldsymbol{\Theta}\right)\\ &-\frac{\lambda}{C-1}\sum_{i=1}^{C-1}Eigen_{val}(\tilde{\Omega}_w u_t t_b u_t^{\top}\tilde{\Omega}_w^{\top})^2\end{aligned} \tag{7.18}$$

Eq. 7.18 has two components: 1) The cross-entropy, which is used to maximize the log-likelihood between mass probability of the label distribution and the predicted class distribution and 2) The regularizer of the joint-nullspace which projects from the feature space to the joint-nullspace. The regularizer of the joint-nullspace maximizes the expected non-zero over Eigenvalues of $(\tilde{\Omega}_w u_t t_b u_t^{\top}\tilde{\Omega}_w^{\top})$ to encourage the network to be more discriminative. Moreover, the regulaizer of the joint-nullspace encourages the network to project the similar target-specific information onto one point and different target-specific information onto different points in the joint-nullspace as discussed above. This behavior is inherited from the two restrictions which are given in Eq. 7.5. Algorithm 5 summarizes the regularized discriminative joint-nullspace framework which is used to train the NullSpace backbone network.

7.4.2 Adaptive Bounding Box Estimation Phase

After the NullSpace backbone is trained by the regularized discriminative loss function in Eq.7.18, the weights of $NullSpaceBlock3$ in Fig. 7.2 are frozen in both branches. Then, the feature maps $B4\_1$ from the training branch are fed into the predictor network to predict the weights of the online classifier filter that discriminates between the target and distractors. The

Algorithm 5: Projection onto a regularized discriminative joint-nullspace

```
input  : ResNet50 backbone where the output of the last block $NullSpaceBlock4\_1$
         $F \in \mathbb{R}^{D \times N}$
output: NullSpace backbone with the features are being projected onto joint-nullspace
1: Forward a batch of images through the backbone network;
2: Find the centric class mean and global mean from $F_w$ and $F_t$;
3: Calculate the between-target-specific scatter matrix, within-target-specific scatter matrix,
   and total scatter matrix from Eq. 7.7 and Eq. 7.9;
4: Calculate the nullspace of the total scatter matrix $\Omega_t$ from Eq. 7.10 and the nullspace of
   the within-target-specific scatter matrix $\Omega_w$ from Eq. 7.11;
5: Factorize the centered output around the global mean using Singular Value Decomposition
   (SVD) from Eq. 7.12;
6: From Eq. 7.15, calculate the nullspace spanned by the reduced version of $\tilde{t}_w$;
7: Find the non-zero eigenvalues $\tilde{\Omega}_w u_t t_b u_t^\top \tilde{\Omega}_w^\top$, and train the network using the regularized
   loss function in Eq.7.18;
```

same predictor network architecture presented in DiMP [10] is used. The feature map $B4\_2$ from the testing branch is fed into the convolutional layer (i.e., the red block in the online classifier) to be convolved with the predicted filter. Consequently, the convolution operation outputs the score map that has a rough estimation of the target bounding box.

The target estimator network uses the proposed novel adaptive super-set loss function to force the network to automatically select the best loss from a family of loss functions. The target estimator network takes the feature maps ($B3\_1$, $B3\_2$, $B4\_1$, and $B4\_2$) from the NullSpace backbone along with the ground truth bounding box and N bounding box proposals to produce the Intersection over Union (IoU) score for each proposal as shown in Fig. 7.2. In particular, the training images are fed into the backbone of NullSpaceRDAR to produce feature maps($B3\_1$ and $B4\_1$)). Then, the target estimator network takes only the first feature map as input from both maps, through two convolutional layers, along with the ground truth bounding box (i.e, the ground truth of the first frame in the training set). The two feature maps are then fed into two differentiable pooling layers (yellow PrPool layer) [39] to output a fixed feature representation for each feature map through two fully connected layers (Green FC layers). Then, the two feature representations are concatenated to produce the reference vector, which is a high-level representation of the target appearance.

Similarly, the test frame is fed into the NullSpace backbone to produce feature maps $B3\_2$ and $B4\_2$ as shown in Fig. 7.2. These feature maps are then fed into the target estimator network through convolutional layers, along with the N bounding box proposals. To generate the bounding box proposals, Gaussian noise is applied to the rough estimation of the bounding box (i.e., the bounding box in the score map) to generate 10 proposals with different scales and translations. The proposals, along with the feature maps, pass through PrPool layers to produce N feature maps (fixed representation for each proposal). The reference vector is fused with the N feature maps of the proposals through channel-wise production. In this proposed design, the target information, which is represented by the reference vector, and the background information, which is represented by the proposals, form the required information about the target and its distractors. The outputs of the channel-wise production are converted into N test vectors by using two fully connected layers as shown in Fig. 7.2. Finally, the test vectors, which have the target and background information, are fed into a fully connected layer to produce the Intersection over Union score (i.e., IoU) between the target bounding box in the training image and the bounding box proposals in the test image.

7.4.2.1 Mathematical Formulation

As shown in Fig. 7.2, the target estimator network is designed to calculate the Intersection over Union (IoU). Let the output of the NullSpaceRDAR target estimator network $n_{iou} \in \mathbb{R}^{\mathbb{N}}$, which is the IoU for each proposal. Moreover, the ground truth IoU of the proposals $p_{iou} \in \mathbb{R}^{\mathbb{N}}$, where N is the number of proposals and p_{iou} is calculated from:

$$p_{iou} = \frac{|B_r \cap B_t|}{|B_r \cup B_t|} \tag{7.19}$$

Where B_r is the bounding box of the target in the training image and B_t is the bounding box of the object in the test image (i.e., the proposal). Note that the proposals in the proposed network are set to be at least 10 proposals and p_{iou} is calculated for each proposal in the test image. Given the IoU ground truth proposals (i.e., p_{iou}) and the output prediction of NullSpaceRDAR target estimator (i.e.,n_{iou}), the objective is to adaptively train the NullSpaceRDAR target estimator network using the super-set loss function. In other words, instead of training the network with the mean squared error L^2 loss that is sensitive to outliers/large errors [33], the adaptive loss function is used. The adaptive loss function automatically switches between loss functions within a super-set of loss functions as described below.

The adaptive bounding box loss function is partially inspired by [33], but our motivation is to make the target estimator network more robust by allowing the network to automatically select the best loss for the bounding box estimation task. The proposed adaptive loss function is a super-set of different loss functions, for example, L^2, L^1, Lorentzian loss function [37], and Charbonnier loss functions [38], to name a few.

The objective in NullSpaceRDAR target estimator network is to maximize the predicted IoU (i.e., n_{iou}). Let the absolute difference between n_{iou} and ground truth proposals p_{iou} be:

$$x = |n_{iou} - p_{iou}| \tag{7.20}$$

The adaptive loss function that maximizes n_{iou} and minimizes the error can be calculated by,

$$L(x, \alpha) = \frac{|\alpha - 2|}{\alpha} e^{ay} \left(\left(\frac{x^2}{|\alpha - 2|} + 1 \right)^{\alpha/2} - 1 \right) \tag{7.21}$$

Where $a \in [0, 1]$ is a hyper-parameter value which can be tuned on the validation set, y is a positive IoU weight vector that can be initialized from a Gaussian distribution. Moreover, $\alpha \in \mathbb{R}$ is the selector parameter that can control the type of loss during training.

It is worth mentioning that different values of α give different loss functions. Specifically, α is a continuous learnable parameter that can capture, for example, L^2, Lorentzian and other in between losses. This means that the adaptive loss can be any of these loss functions during the optimization. For example, setting $\alpha = 2$, the loss function will approach L^2,

$$\lim_{\alpha \to 2} L(x, \alpha) = \frac{e^{ay}}{2} x^2 \tag{7.22}$$

The loss function becomes a Lorentzian loss function when $\alpha = 0$,

$$\lim_{\alpha \to 0} L(x, \alpha) = \log \left(\frac{1}{2} x^2 + 1 \right) e^{ay} \tag{7.23}$$

By setting $\alpha = 1$, the loss function turns into a smoothed L^1,

$$L(x,\alpha) = \left(\sqrt{x^2+1}\right) e^{ay} - 1 \tag{7.24}$$

The loss function based on the different α values has four bases as follows,

$$\mathbb{L}_{est}(x,\alpha) = \begin{cases} \frac{e^{ay}}{2} x^2 & \text{if } \alpha = 2 \\ \log\left(\frac{1}{2}(x)^2 + 1\right) e^{ay} & \text{if } \alpha = 0 \\ \left(1 - \exp\left(-\frac{1}{2}(x)^2\right)\right) e^{ay} & \text{if } \alpha = -\infty \\ \frac{|\alpha-2|}{\alpha} e^{ay} \left(\left(\frac{(x)^2}{|\alpha-2|} + 1\right)^{\alpha/2} - 1\right) & \text{otherwise} \end{cases} \tag{7.25}$$

It can be seen from Eq. 7.25 that the loss function always increases monotonically w.r.t. α due to $\frac{\partial \mathbb{L}}{\partial \alpha}(x,\alpha) \geq 0$. The gradient of the loss function in Eq. 7.25 is calculated as follows,

$$\frac{\partial \mathbb{L}_{est}}{\partial x}(x,\alpha) = \begin{cases} x e^{ay} & \text{if } \alpha = 2 \\ \frac{2x e^{ay}}{x^2+2} & \text{if } \alpha = 0 \\ x e^{\left(ay - \frac{1}{2}x^2\right)} & \text{if } \alpha = -\infty \\ x e^{ay} \left(\frac{x^2}{|\alpha-2|} + 1\right)^{\frac{\alpha-2}{2}} & \text{otherwise} \end{cases} \tag{7.26}$$

For the sake of the optimization, we take the negative log-likelihood of the probability density function defined over the adaptive loss function of Eq. 7.25 as follows,

$$p(x|\mu,\alpha) = \frac{1}{S(\alpha)} exp(-\mathbb{L}_{est}) \tag{7.27}$$

Where μ is distribution mean, p is the probability density function of the adaptive loss, and (α) is the scale function defined by $S = LeakyRelu(\alpha)$. Consequently, the negative log-likelihood is given by,

$$-log(p(x|\mu,\alpha)) = \mathbb{L}_{est} + log(S(\alpha)) \tag{7.28}$$

Where α now is a free and differentiable parameter that is set by NullSpaceRDAR during the training. To initialize α, a scaled version of the Sigmoid function is used as follows,

$$\alpha = \frac{2}{e^{-i} + 1} \tag{7.29}$$

Where $i \in [0, 2]$, and Sigmoid to keep the value in range $[0, 2]$. In this setting, NullSpaceRDAR decides the degree or robustness of the adaptive loss function.

7.5 NullSpaceRDAR Offline Training

NullSpaceRDAR is an end-to-end architecture, which means that the whole architecture, including the online classifier and target estimator network, is trained offline. The online classifier uses a combination of L^2 and hinge loss depending on the location of the target in the image. The loss function of the online classifier is given by,

$$L_{cls} = \frac{1}{n} \sum_{n} \| h - y_c) \|^2 + \lambda \| w \|^2 \tag{7.30}$$

Where n is the batch size and h is the output weighted score map (s), as shown in Fig. 7.2, by a Gaussian kernel g_c to avoid imbalanced classes. In this network, y_c is the ground truth score map which can be initialized from a Gaussian distribution with a high score at the target location and a low score in the background region. The weighted score map h is given by,

$$h = \begin{cases} g_c s & \text{if target region} \\ g_c * Relu(s) & \text{if background region} \end{cases} \tag{7.31}$$

After the projection onto the joint-nullspace, NullSpaceRDAR freezes the weights of $NullSpaceBlock3\_1$ and $NullSpaceBlock3\_2$ in both branches of the backbone network to fully exploit the power of the features in the regularized joint-nullspace during the training. The NullSpaceRDAR target estimator network is simultaneously trained by using the negative log-likelihood as shown in Eq. 7.28. At the beginning of the training, α is initialized using Eq.7.29, and during the training optimization, NullSpaceRDAR is free to select the best possible loss function from the super-set. For example, during the training optimization, NullSpaceRDAR may select L^2 loss when the α approaches 2. Similarly, NullSpaceRDAR may select L^1 when α approaches 1. In this setting, different values of α have different robustness. For example, L^1 tolerates large errors compared to L^2, which is sensitive to outliers/large errors. The unified loss function which is used to train NullSpaceRDAR is given by,

$$L_{unified} = \gamma L_{cls} - \beta log(p(x|\mu, \alpha)) \tag{7.32}$$

Where γ and β are hyper-parameters that can be selected based on the validation set.

7.6 NullSpaceRDAR Online Tracking

In online tracking, the first frame is augmented by applying flipping, channel-wise dropout, Gaussian noise with $\sigma = 0.01$ to blur out the frame, shifting, and rotation. These augmented frames result in 16 frames acting as a training set. Then, the augmented frames are fed into the training branch of NullSpaceRDAR backbone to produce 16 feature maps from $NullSpaceBlock4\_1$ in the training branch as shown in Fig. 7.2. The feature maps are then fed into the online classifier, which has the predictor network. The predictor consists of an initializer network and optimizer. The initializer network has a convolutional layer followed by a PrPool layer [39] which is used to produce an initial estimate of the filter's weights. The optimizer refines the initial estimate of the filter's weights using steepest descent to calculate the final filter (i.e.,the filter that has the target-specific and background-specific information) of the classifier. The online classifier is used to distinguish the target object from distractors (i.e., the background and other objects). The test frame (i.e., starting from the second frame), is fed into the test branch, the NullSpace feature extractor backbone produces a feature map from $NullSpaceBlock4\_2$, that is for a single test frame. This feature map is then fed into the online classifier through a convolution layer which convolves the feature map by the predicted filter, producing a score map. It is worth mentioning that the score map gives an initial estimation of the target location. Once NullSpaceRDAR has the initial target location (i.e., the bounding box) from the classifier module, it applies a Gaussian noise to the target center to generate 10 candidate proposals for the task of accurate target estimation. The target estimator network produces an IoU score for each proposal as detailed in sub-section 7.4.2 by overlap maximization. The final target location is then calculated by the mean of the highest of the three IoU scores.

7.7 Experimental Results

Datasets:

NullSpaceRDAR has been trained on five different datasets to provide extensive results and guarantee more diversity of classes. Specifically, we used ImageNet [30] for feature projection onto the joint-nullspace and the training split of GOT10k [40], LaSOT [41], COCO [42], and TrackingNet [43] for the adaptive bounding box estimation phase.

1) ImageNet: This dataset has different sequences for different tasks such as object detection and image classification. ImageNet has 4000 sequences and 1.3 million images for object classification. NullSpaceRDAR has been trained on ImageNet to project the input onto the proposed joint-nullspace.

2) GOT10k: Contains natural images of 1.5 million labeled images. GOT10k is a large-scale dataset with over 560 classes of real-world moving objects as well as over 80 classes for motion patterns.

3) LaSOT: It stands for Large-Scale Single Object Tracking which has varieties of different classes. LaSOT contains natural images for 70 categories, each category contains 20 sequences. Moreover, LaSOT has long-term videos to validate long-term trackers.

4) COCO: This is another large-scale dataset that has a variety of sequences for different vision tasks such as object segmentation, object recognition, and object detection. COCO has 80 categories and over 200,000 frames with their labels for the task of object detection.

5) TrackingNet: This is a large-scale dataset for object tracking in the wild. It has over 30,000 sequences associated with 14 million bounding boxes. The majority sequences in the TrackingNet dataset are chosen in the wild to cover more real-world scenarios such as occlusions and scale variations.

6) OTB benchmark: This benchmark has different challenging attributes with three sets, namely OTB50, OTB100, and OTB2013. In our experiments, we used OTB100 which has 100 sequences.

7) VOT benchmark: VOT is considered to be the golden testing benchmark in visual object tracking. It is usually held every year with/without some changes in the dataset or the evaluation toolkit. It has 60 sequences collected from natural images. We used VOT2015, VOT2016, VOT2018, and VOT2019 for testing.

Evaluation metrics:

To evaluate NullSpaceRDAR, standard evaluation metrics from the VOT challenges [11] are adopted. In particular, Accuracy (A), Robustness (R), and Expected Average overlap (EAO). Moreover, the precision and success metrics are used to report the performance of NullSpaceR-

DAR on the OTB datasets [44].

Accuracy (A): is the average of the Intersection over Union (IoU) in successful frames. The accuracy is only measured after 10 frames from the re-initialization process.

Expected Average Overlap (EAO): is the average overlap between the predicted bounding box and the ground truth bounding box. The average is calculated over all test sequences.

Robustness(R): is the measure of how many times the tracker fails to track the target object. The failure of the tracker is reported when the Intersection over Union (IoU) is zero.

Precision: measures the percentage of frames whose predicted center is within 20 pixel from the ground truth center [11].

Success Plot: is the ratio of successful frames at a range of thresholds (i.e. from 0 to 1).

7.7.1 Implementation details

NullSpaceRDAR has been implemented with Python and Pytorch framework [80]. The backbone network has been defined in a mixed-precision using the Nvidia Apex library [81]. NullSpaceRDAR has been trained using 4 GPUs (Nvidia Tesla V100) with each containing 32 GB of memory. The source code is implemented on a single machine with multi-GPU with data parallelism to take advantage of all available resources. A Xeon E5 CPU (2.20 GHz) and 32 GB of RAM are powering the machine, which runs a Linux operating system. ResNet50 is initialized by ImageNet weights and trained using the regularized adaptive discriminative loss function to project the feature space onto the joint-nullspace. The training is done on 50 epochs and 20.000 sequences per epoch. The online classifier is optimized for 5 iterations and the training batch is split into 3 frames for training and 3 for testing from a segment of length 60. The predicted filter (i.e., the filter in the online classifier) is set to 4×4 and the learning rate starts from 10^{-2}

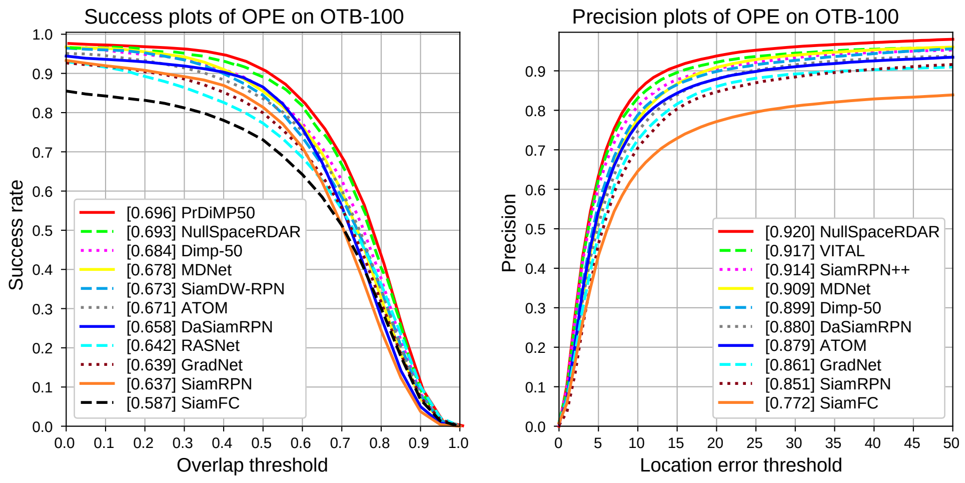


Figure 7.5: The precision plots and success plots of OTB benchmarks.

and is annealed geometrically to 10^{-5}. The number of proposals is set to 10, extracted from a Gaussian noise with $\sigma^2 = 0.2$.

7.7.2 Comparison with the state-of-the-arts

NullSpaceRDAR is tested and compared to state-of-the-art trackers on different datasets such as OTB [44], VOT [11], UAV123 [45], and the testing split of LaSOT [41], TrackingNet [43], and GOT10k [40]. None of the trackers were modified to give NullspaceRDAR an advantage. All trackers were used as-is to maintain a fair comparison and avoid any potential bias or mistakes.

Table 7.1: Comparison of state-of-the-art real-time trackers on OTB benchmarks.

Dataset	Metric	NullSpaceRDAR (proposed)	PrDiMP50 [29]	PrDiMP18 [29]	DiMP50 [10]	DiMP18 [10]	DPSiam [8]	BACF [46]	PTAV [47]	ECOhc [48]	DSiamM [49]	EAST [50]	Staple [51]	SiamFC [13]
OTB100	AUC	0.693②	0.696①	0.680③	0.684	0.660	0.677	0.621	0.635	0.643	-	0.629	0.578	0.582
	Prec.	0.920①	-	-	0.899②	0.865	0.883③	0.822	0.849	0.856	-	-	0.784	0.771

Table 7.2: Comparison with the state-of-the-art trackers on VOT2015 dataset using standard metrics Accuracy (A), Expected Average Overlap (EAO), and Robustness (R).

Tracker	A↑	EAO↑	R↓	FPS
DeepSRDCF [52]	0.56	0.32	1.05	10
EBT [53]	0.47	0.31	1.05	4
SRDCF [54]	0.56	0.29	1.24	5
HP[55]	0.58	0.33	1.578	69
RAJSSC [56]	0.57	0.34	1.63	22
MDNet [57]	0.60②	0.38②	0.69②	1
OACF [58]	0.58	0.36③	1.81	5
BACF [46]	0.59	-	1.56	35
EAST [50]	0.57	0.34	1.03	159
Staple [51]	0.57	0.30	1.39	80
SiamFC [13]	0.55	0.29	1.58	86
SiamRPN [5]	0.59③	0.35	0.93③	160
NullSpaceRDAR(proposed)	0.62 ①	0.440①	0.68①	41

7.7.2.1 OTB benchmark

Table 7.1 shows the performance in terms of Area Under Curve (AUC) and precision on OTB100. As shown in Table 7.1, NullSpaceRDAR achieves the second-best tracker in terms of AUC with 0.693, while the top tracker is PrDiMP50 with 0.696. The PrDiMP tracker achieves a gain of $+0.3$ compared to NullSpaceRDAR. It is worth mentioning that PrDiMP has two versions, PrDiMP50 which has ResNet50 as a backbone network and PrDiMP18 which has ResNet18

Table 7.3: Comparison with *state-of-the-art* trackers on VOT2016 dataset using standard metrics Accuracy (A), Expected Average Overlap (EAO), and Robustness (R).

Tracker	A↑	EAO↑	R↓	FPS
MDNet_N [57]	0.541	0.257	0.337	1
SSAT [59]	0.577③	0.321	0.291	22
SiamFC [13]	0.532	0.235	0.461	86
HP [55]	0.539	0.242	0.46	69
STAPLE+ [59]	0.557	0.286	0.368	25
SRBT [60]	0.496	0.290	0.350	15
NSAMF [61]	0.502	0.227	0.438	25
DPT [62]	0.492	0.236	0.489	10
ColorKCF [59]	0.503	0.226	0.443	90
GCF [59]	0.520	0.218	0.485	7
SiamRPN[5]	0.560	0.344③	0.260③	160
SiamRPN++[7]	0.640②	0.464②	0.200②	-
NullSpaceRDAR(proposed)	0.690①	0.530①	0.180①	41

as a backbone network. PrDIMP18 achieves 0.680 in terms of AUC, which is ranked as the third-best tracker. On the other hand, NullSpaceRDAR achieves the best precision of 0.920 with gain of $+2.1\%$ compared to the second-best tracker DiMP with precision of 0.899. It clearly shows that NullSpaceRDAR outperforms all other trackers in terms of precision with a significant margin. It worth noting that, PrDiMP50, PrDiMP18, and DiMP50 share the ResNet network architecture in the feature space while NullSpaceRDAR uses the Nullspace backbone

Table 7.4: Comparison with *state-of-the-art* trackers on VOT2018 dataset using standard metrics Accuracy (A), Expected Average Overlap (EAO), and Robustness (R).

Tracker	A↑	EAO↑	R↓	FPS
FSAN [9]	0.554	0.256	0.356	30
CSRDCF [54]	0.491	0.256	0.356	13
ASMS [63]	0.494	0.169	0.623	25
SiamRPN [5]	0.586	0.383	0.276	160
SA_Siam_R [64]	0.566	0.337	0.258	50
SiamFC [13]	0.503	0.188	0.585	86
SAPKLTF [9]	0.488	0.171	0.613	25
DSiam [49]	0.215	0.196	0.646	25
ECO [48]	0.484	0.280	0.276	60
ATOM [12]	0.590	0.401	0.204	-
SiamRPN++ [7]	0.600	0.414	0.234	-
DiMP18 [10]	0.594	0.402	0.182	-
DiMP50 [10]	0.597	0.440③	0.153②	-
PrDiMP18 [29]	0.607③	0.385	0.217	-
PrDiMP50 [29]	0.618②	0.442②	0.165③	-
NullSpaceRDAR(proposed)	0.627①	0.446①	0.140①	41

network in the joint-nullspace. The plot in Fig. 7.5 shows the success plot and precision plot on OTB100, the results are given by using the OTB toolkit [44] on the publicly available source codes.

Table 7.5: Comparison with *state-of-the-art* trackers on VOT2019 dataset using standard metrics Accuracy (A), Expected Average Overlap (EAO), and Robustness (R).

Tracker	A↑	EAO↑	R↓	FPS
ATOM [12]	0.603①	0.292	0.411	-
SiamRPN++ [7]	0.599②	0.285	0.482	-
DiMP50 [10]	0.594③	0.379②	0.278②	-
SiamMask [6]	0.594	0.287	0.461	-
MemDTC [65]	0.485	0.228	0.587	-
STN [66]	0.589	0.314	0.349	-
Ocean [67]	0.594	0.350	0.316③	-
DCFST [68]	0.589	0.361③	0.321	-
NullSpaceRDAR(proposed)	0.603①	0.384①	0.250①	41

7.7.2.2 VOT benchmark

Table 7.2 shows the results of the proposed tracker (NullSpaceRDAR) on VOT2015 compared to the top trackers in VOT2015 [59]. As shown in Table 7.2, NullSpaceRDAR outperforms all trackers. In terms of Accuracy (A) NullSpaceRDAR achieves a score of 0.62 with a gain of $+2\%$ compared to the second-best tracker MDNet. It is worth mentioning that NullSpaceRDAR achieves the best rank in terms of Expected Average Overlap (EAO) with a significant margin of $+6\%$ compared to the second-best tracker MDNet. In term of Robustness (R), NullSpaceRDAR achieves the best performance with a score of 0.68 compared to the second-best tracker MDNet.

In VOT2016 [78], the ground truth labels have been re-labeled and each pixel has been segmented compared to VOT2015. Table 7.3 shows the results of NullSpaceRDAR compared to

Table 7.6: Comparisons with *state-of-the-art* trackers on TrackingNet dataset in terms of the Precision (PRE), Normalized Precision (NPRE), and Success.

Tracker	PRE ↑	NPRE ↑	SUC.↑
Staple_CA [69]	0.468	0.605	0.529
BACF [46]	0.461	0.580	0.523
MDNet [57]	0.565	0.705	0.606
CFNet [14]	0.533	0.654	0.578
SiamFC [13]	0.533	0.663	0.571
SAMF [61]	0.477	0.598	0.504
ECO-HC [48]	0.476	0.608	0.541
Staple [51]	0.470	0.603	0.528
ECO [48]	0.492	0.618	0.554
CSRDCF [54]	0.480	0.622	0.534
ATOM [12]	0.648③	0.771	0.703
DiMP18 [10]	0.666②	0.785③	0.723③
DiMP50 [10]	0.687①	0.801②	0.740②
NullSpaceRDAR(proposed)	0.687 ①	0.813①	0.749 ①

the top trackers in VOT2016. NullSpaceRDAR outperforms all trackers in terms of Accuracy (A) with a score of 0.690 compared to the second-best tracker SiamRPN++. In terms of EAO, NullSpaceRDAR achieves state-of-the-art with a score of 0.530 and the second-best tracker SiamRPN++ achieves 0.464. Similarly, NullSpaceRDAR is ranked the best tracker in terms of Robustness (R) with a score of 0.180 while the second-best tracker SiamRPN++ has a score of

Table 7.7: Comparison with *state-of-the-art* trackers on GOT10k dataset in terms of Average Overlap (AO), and Success Rates (SR) at overlap thresholds of 0.50 and 0.75.

TRACKER	NullSpaceRDAR	ATOM [12]	DiMP50[10]	DomainSiam [70]	CFNet [14]	SiamFC [13]	GOTURN	CCOT [71]	ECO [48]	HCF	MDNet [57]
AO	0.626①	0.556③	0.611②	0.414	0.374	0.348	0.347	0.325	0.316	0.315	0.299
SR(0.50)	0.726①	0.634③	0.717②	0.451	0.404	0.353	0.375	0.328	0.309	0.297	0.303
SR(0.75)	0.498①	0.402③	0.492②	0.214	0.144	0.098	0.124	0.107	0.111	0.088	0.099

Table 7.8: Comparison with *state-of-the-art* trackers on LaSOT dataset in terms of the Normalized Precision and Success.

Tracker	Norm. Prec. (%)↑	Success (%)↑
MDNet [57]	46.0	39.7
DaSiam [72]	49.6	41.5
STRCF [73]	34.0	30.8
SINT [74]	35.4	31.4
StrucSiam [75]	41.8	33.5
SiamFC [13]	42.0	33.6
VITAL [76]	45.3	39.0
ECO [48]	33.8	32.4
DSiam [77]	40.5	33.3
ATOM [12]	57.6③	51.5③
SiamRPN++ [7]	56.9	49.5
DiMP50 [10]	64.3②	56.9②
NullSpaceRDAR(proposed)	65.7①	57.4①

0.20. It is worth noting that NullSpaceRDAR outperforms all tracker in VOT2016 while working in real-time speed of 41 FPS. It is worth mentioning that VOT2018 includes all sequences from VOT2017.

In VOT2018 [9], a long-term tracking challenge has been introduced (does not apply to the proposed tracker) along with minor changes in the dataset bounding boxes. As shown in Table 7.4, NullSpaceRDAR is ranked the best tracker in terms of Accuracy (A) with a score of 0.627 with a gain of $+0.9$ compared to the second-best tracker PrDiMP50 with a score of 0.618. NullSpaceRDAR achieves a score of 0.446 in terms of EAO and comes in the first place before PrDiMP50 with a score of 0.442. In terms of Robustness (R) NullSpaceRDAR comes in the first place with a gain of $+1.3$ compared to DiMP50. Even though NullSpaceRDAR is built in DiMP framework, it outperforms all variations of DiMP trackers thanks to the joint-nullspace and the adaptive loss function.

In VOT2019 [11], the dataset has been updated along with introducing two new challenges in thermal and depth imagery (do not apply to our proposed tracker). As shown in table 7.5, NullSpaceRDAR is ranked as the top tracker in terms of accuracy with a score of 0.603. While ATOM achieves a similar accuracy of 0.603, however, in terms of EAO, NullSpaceRDAR achieves 0.384, which is a leading score with a gain of $+9.2\%$ compared to ATOM. Moreover, NullSpaceRDAR outperforms DiMP50, the second-best tracker, with a gain of $+0.5$ in terms of EAO. Similarly, NullSpaceRDAR outperforms all other trackers in terms of robustness with a score of 0.250 while DiMP50 achieves the second-best tracker with a score of 0.278. The visual results on VOT2019 are shown in Fig. 7.7

These results on VOT benchmark confirm that NullSpaceRDAR outperforms all current state-of-the-art trackers on VOT benchmark.

7.7.2.3 TrackingNet

As illustrated in Table 7.6, precision, normalized precision, and success rate are used to evaluate the trackers in this dataset. Precision measures the percentage of frames whose predicted center is within a threshold from ground truth center. Normalized precision is introduced to overcome the sensitivity of precision to the resolution of the images, the precision is normalized over the size of the ground truth bounding box [43]. The success rate measures the percentage of frames where the predicted bounding box overlaps with the ground truth within a threshold. Table 7.6 shows that NullSpaceRDAR has the best precision (PRE) score of 0.687 (tied with DiMP50). In terms of the Normalized precision (NPRE), NullSpaceRDAR is ranked the best tracker with a gain of $+1.2$ over DiMP50. The Success (SUC) of NullSpaceRDAR is 0.749 with a gain of $+0.9$ compared to DiMP50. Since TrackingNet is a large-scale dataset that has varieties of sequences and objects, this confirms our claim of the generalization ability of NullSpaceRDAR.

7.7.2.4 GOT10k

The proposed tracker was further evaluated on the GOT10k [40] dataset. Table 7.7 shows the comparison between NullSpaceRDAR and state-of-the-art trackers ATOM [12], DiMP [10], DomainSiam [70], CFNet [14], SiamFC [13], GOTURN [79], CCOT [71], ECO [48], and MDNet [57]. It is clear that NullSpaceRDAR achieves a leading performance on the large-scale dataset GOT10k. In particular, NullSpaceRDAR achieves the best Average Overlap (AO) with a score of 0.626 and the second-best tracker DiMP achieves 0.611, this gives a gain of $+1.5\%$ in favor of NullSpaceRDAR. GOT10k evaluates the Success Rate (SR) at 0.50 and 0.75. In terms of $SR = 0.50$, NullSpaceRDAR is ranked the best tracker with a score of 0.726 and the second-best tracker DiMP50 achieves 0.717. In terms of $SR = 0.75$, NullSpaceRDAR outperforms

DiMP50 with gain of $+0.6$ in favor of NullSpaceRDAR. GOT10k is another large-scale with high diversity classes, it has over 10,000 videos, this another evidence for the efficient of the joint-nullspace along with the adaptive loss function.

7.7.2.5 LaSOT

LaSOT is another large-scale dataset. Table 7.8 shows the results of NullSpaceRDAR along with state-of-the-art trackers, MDNet [57], DaSiam [72], STRCF [73], SINT [74], StrucSiam [75], SiamFC [13], VITAL [76], ECO [48], DSiam [77], ATOM [12], SiamRPN++ [7], and DiMP [10]. As shown in Table 7.8 NullSpaceRDAR achieves the best performance in terms of Normalized Precision (Norm Prec.) with a score of 65.7%, while DiMP50 comes in the second-best rank with a score of 64.3%. Similarly, NullSpaceRDAR outperforms DiMP50 with again of $+0.5$ in favor of NullSpaceRDAR in terms of success rate. This affirms the claim that the NullSpace backbone affects the final performance of the tracker and significantly increases the performance on large-scale datasets.

7.7.2.6 Individual Challenging Attributes

To further evaluate NullSpaceRDAR, another experiment has been conducted on twelve challenging attributes from the UAV123 [45] dataset. The twelve attributes contain viewpoint change, out-of-view, low resolution, background clutter, camera motion, similar object, scale variation, illumination variation, fast motion, full occlusion, partial occlusion, and aspect-ratio changes. Fig. 7.6 shows the success plots on UAV123. Out of twelve challenging attributes, NullSpaceRDAR outperforms all other trackers on 10 out of the 12 challenging attributes. Moreover, NullSpaceRDAR is ranked the second-best tracker on the remainder 2 attributes, the scale-variation and aspect-ratio attributes. The reason behind that is that NullSpaceRDAR does

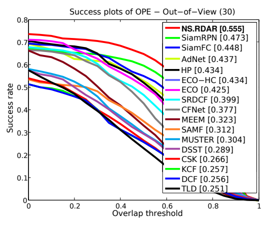
Table 7.9: Ablation study using variations of NullSpaceRDAR

Variations of DP-Siam			LaSOT		OTB100	
Baseline	Joint-nullspace	Adaptive B.B. loss	Norm. Prec	Success	AUC	Prec.
✓			0.643	0.569	0.684	0.899
✓		✓	0.649	0.571	0.687	0.906
✓	✓		0.653	0.572	0.690	0.918
✓	✓	✓	0.657	0.574	0.693	0.920

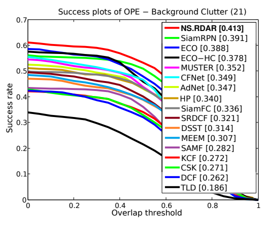
not use the bounding box regression to regress the coordinates. NullSpaceRDAR is anchor-free, which uses bounding box estimation to refine the proposals that were generated by Gaussian noise on the roughly calculated bounding box center from the online classifier.

7.7.3 Ablation Study

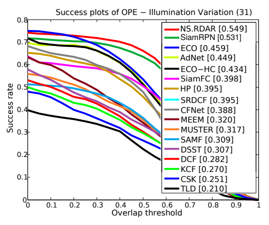
To show the impact of each component in the proposed NullSpaceRDAR, Table 7.9 shows an ablation study on variations of NullSpaceRDAR. The baseline in Table 7.9 means NullSpaceRDAR without the joint-nullspace (i.e., feature space) and adaptive bounding box estimation loss (i.e., mean squared loss). Specifically, adding just the adaptive bounding box estimation loss will boost the performance of the tracker from 0.643 in terms of normalized precision to 0.649 on LaSOT dataset. Similarly, the adaptive bounding box estimation loss will boost the performance of NullSpaceRDAR on OTB100 from 0.684 in terms of AUC to 0.687 and the precision from 0.899 to 0.906. On the other hand, the projection from the feature space to the joint-nullspace boosts the performance of NullSpaceRDAR from 0.643 to 0.653 in terms of normalized precision with a gain of 1% compared to $+0.6\%$ in case of adaptive bounding box



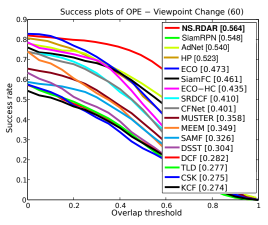
(a)



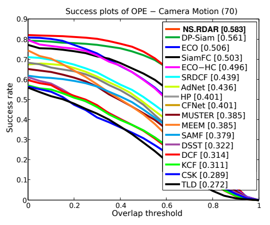
(b)



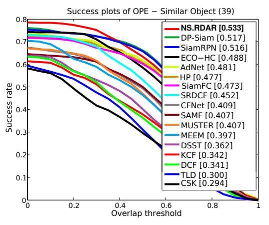
(c)



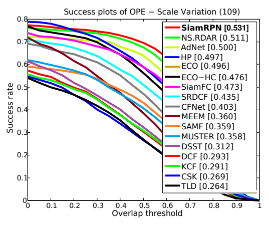
(d)



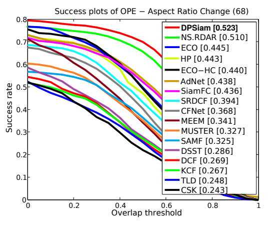
(e)



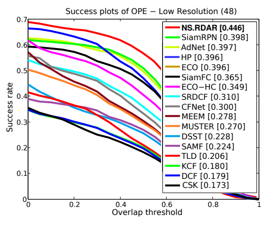
(f)



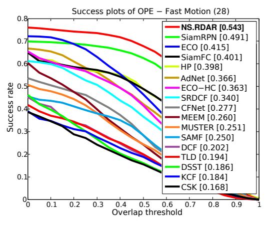
(g)



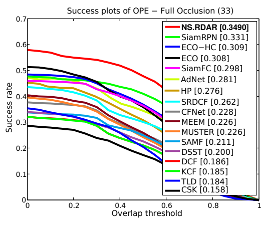
(h)



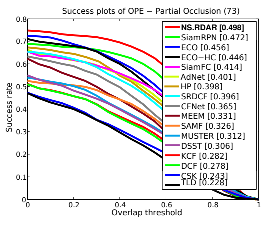
(i)



(j)

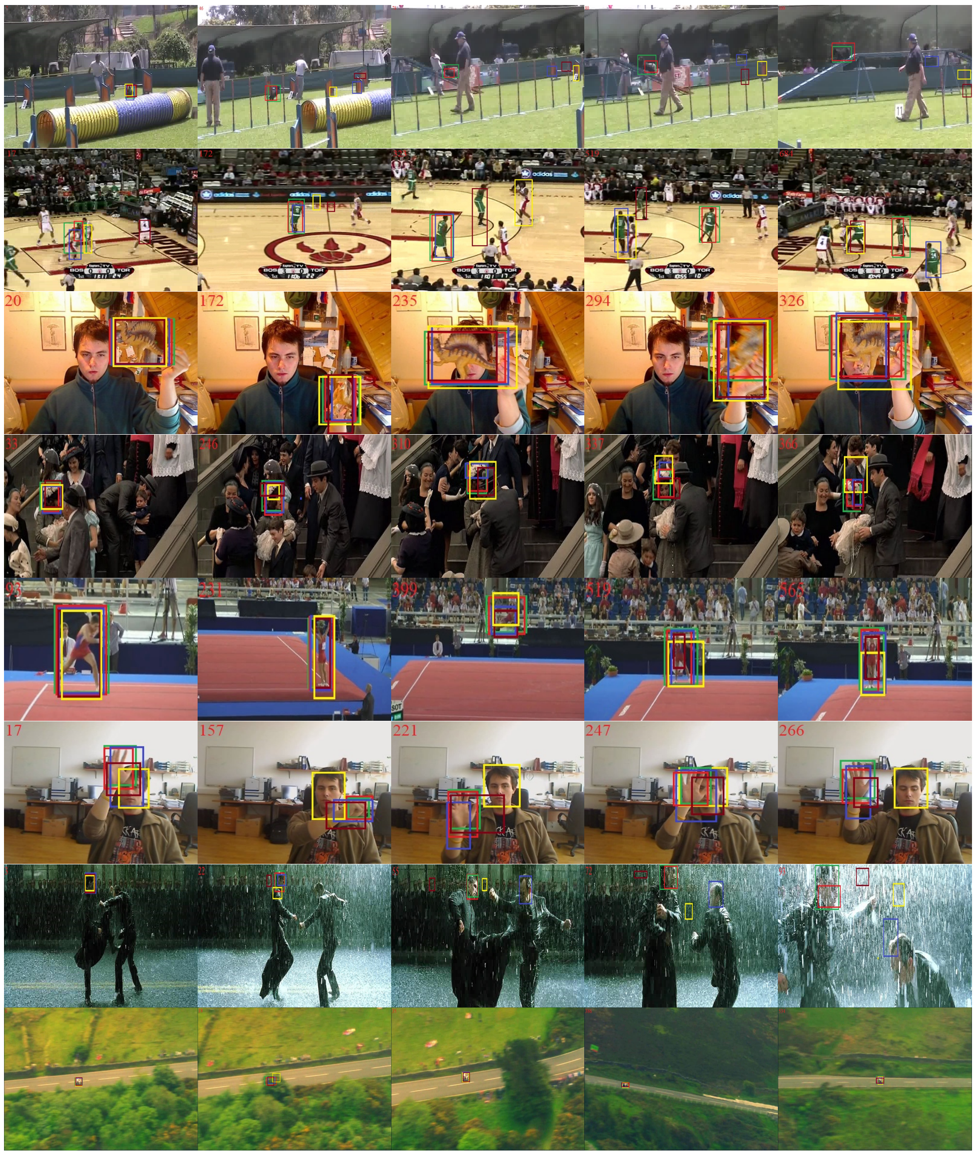


(k)



(l)

Figure 7.6: Success plots on UAV123 over 12 attributes. Note that due to the limit of space, NS.RDAR is the shortened form of the proposed tracker (NullSpaceRDAR).



–Ground truth –NullSpaceRDAR(Ours) –DiMP –ATOM –SiamRPN

Figure 7.7: Visual results for the VOT2019 dataset on different sequences. The sequences are ordered in rows while the output of the trackers are ordered in columns. The frame number is printed on the top left corner. From the top row to the bottom row, the sequences are agility, basketball, dinosaur, godfather, gymnastics1, hand, matrix, and road.

estimation loss. Similarly, on OTB100, adaptive bounding box estimation loss will boost the AUC by $+0.6$ and the precision from 0.899 to 0.918. Adding all components (i.e., the joint-nullspace and the adaptive bounding box estimation loss) will boost the overall performance on LaSOT by 1.4% in terms of normalized precision, 0.5% in terms of success, 0.9% in terms of AUC on OTB100, and 1.9% in terms of precision on OTB100. It is clear from Table 7.9 that the larger impact is from the joint-nullspace. Consequently this supports the effect of the joint-nullspace.

7.8 Conclusions and Future Work

In this paper, a novel regularized discriminative adaptive joint-nullspace tracker is proposed which tracks the objects of interest. The joint-nullspace ensures that the features from the same target-specific information are collapsed into one point and the different features are collapsed into different points in the joint-nullspace. The joint-nullspace has a strong discriminative power due to its formulation, which increases between-target-specific scatter matrix and decreases the within-target-specific scatter matrix. Moreover, to accurately locate the objects of interest, an adaptive bounding box estimation loss function is proposed in the context of visual object tracking to select the best loss function from a super-set family. NullSpaceRDAR is trained in an end-to-end manner and significantly increases the robustness and the generalization of the tracker. The future work includes formulating the online classifier and the target estimator network in one network using the joint-nullspace.

References

[1] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu, "Joint monocular 3d vehicle detection and tracking," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 5390–5399.

[2] H. Qi, C. Feng, Z. Cao, F. Zhao, and Y. Xiao, "P2b: Point-to-box network for 3d object tracking in point clouds," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[3] X. Chen, X. Yan, F. Zheng, Y. Jiang, S.-T. Xia, Y. Zhao, and R. Ji, "One-shot adversarial attacks on visual tracking with dual attention," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[4] N. Scheiner, F. Kraus, F. Wei, B. Phan, F. Mannan, N. Appenrodt, W. Ritter, J. Dickmann, K. Dietmayer, B. Sick, and F. Heide, "Seeing around street corners: Non-line-of-sight detection and tracking in-the-wild using doppler radar," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8971–8980.

[6] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," *arXiv preprint arXiv:1812.05050*, 2018.

[7] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese

visual tracking with very deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4282–4291.

[8] M. H. Abdelpakey and M. S. Shehata, "Dp-siam: Dynamic policy siamese network for robust object tracking," *IEEE Transactions on Image Processing*, vol. 29, pp. 1479–1492, 2019.

[9] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey *et al.*, "The sixth visual object tracking vot2018 challenge results," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[10] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6182–6191.

[11] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. Cehovin Zajc, O. Drbohlav, A. Lukezic, A. Berg *et al.*, "The seventh visual object tracking vot2019 challenge results," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[12] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Atom: Accurate tracking by overlap maximization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4660–4669.

[13] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*. Springer, 2016, pp. 850–865.

[14] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. Torr, “End-to-end representation learning for correlation filter based tracking,” in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 5000–5008.

[15] M. M. Mohamed, “Denssiam: End-to-end densely-siamese network with self-attention model for object tracking,” in *Advances in Visual Computing: 13th International Symposium, ISVC 2018, Las Vegas, NV, USA, November 19–21, 2018, Proceedings*, vol. 11241. Springer, 2018, p. 463.

[16] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[17] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[19] P. Voigtlaender, J. Luiten, P. H. Torr, and B. Leibe, “Siam r-cnn: Visual tracking by re-detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[20] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, “Siamese box adaptive network for visual tracking,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[21] D. Guo, J. Wang, Y. Cui, Z. Wang, and S. Chen, "Siamcar: Siamese fully convolutional classification and regression for visual tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[22] E. Park and A. C. Berg, "Meta-tracker: Fast and robust online adaptation for visual object trackers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 569–585.

[23] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," 2016.

[24] T. Munkhdalai and H. Yu, "Meta networks," *Proceedings of machine learning research*, vol. 70, p. 2554, 2017.

[25] K. Dai, Y. Zhang, D. Wang, J. Li, H. Lu, and X. Yang, "High-performance long-term tracking with meta-updater," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[26] G. Wang, C. Luo, X. Sun, Z. Xiong, and W. Zeng, "Tracking by instance detection: A meta-learning approach," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[27] J. Choi, J. Kwon, and K. M. Lee, "Deep meta learning for real-time target-aware visual tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[28] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[29] M. Danelljan, L. V. Gool, and R. Timofte, “Probabilistic regression for visual tracking,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[31] D. H. Foley and J. W. Sammon, “An optimal set of discriminant vectors,” *IEEE Transactions on computers*, vol. 100, no. 3, pp. 281–289, 1975.

[32] V. G. Jensen, H. Jenson, and G. Jeffreys, *Mathematical methods in chemical engineering*. Elsevier, 1977.

[33] J. T. Barron, “A general and adaptive robust loss function,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4331–4339.

[34] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[35] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, “Image clustering using local discriminant models and global integration,” *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2761–2773, 2010.

[36] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[37] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer vision and image understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[38] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Proceedings of 1st International Conference on Image Processing*, vol. 2. IEEE, 1994, pp. 168–172.

[39] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 784–799.

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

[41] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, "Lasot: A high-quality benchmark for large-scale single object tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 5374–5383.

[42] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

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

[44] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[45] M. Mueller, N. Smith, and B. Ghanem, “A benchmark and simulator for uav tracking,” in *European conference on computer vision*. Springer, 2016, pp. 445–461.

[46] H. Kiani Galoogahi, A. Fagg, and S. Lucey, “Learning background-aware correlation filters for visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[47] H. Fan and H. Ling, “Parallel tracking and verifying: A framework for real-time and high accuracy visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5486–5494.

[48] M. Danelljan, G. Bhat, F. S. Khan, M. Felsberg *et al.*, “Eco: Efficient convolution operators for tracking.” in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[49] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, “Learning dynamic siamese network for visual object tracking,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1763–1771.

[50] C. Huang, S. Lucey, and D. Ramanan, “Learning policies for adaptive tracking with deep feature cascades,” in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 105–114.

[51] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, “Staple: Complementary learners for real-time tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1401–1409.

[52] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, “Convolutional features for correlation filter based visual tracking,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 58–66.

[53] G. Zhu, F. Porikli, and H. Li, “Tracking randomly moving objects on edge box proposals,” *arXiv preprint arXiv:1507.08085*, 2015.

[54] A. Lukezic, T. Vojir, L. C. Zajc, J. Matas, and M. Kristan, “Discriminative correlation filter with channel and spatial reliability.” in *CVPR*, vol. 6, 2017, p. 8.

[55] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, and F. Porikli, “Hyperparameter optimization for tracking with continuous deep q-learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 518–527.

[56] M. Zhang, J. Xing, J. Gao, X. Shi, Q. Wang, and W. Hu, “Joint scale-spatial correlation tracking with adaptive rotation estimation,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 32–40.

[57] H. Nam and B. Han, “Learning multi-domain convolutional neural networks for visual tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[58] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr, “The importance of estimating object extent when tracking with correlation filters,” in *Report for the Visual Object Tracking Workshop*, 2015.

[59] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, “The visual object tracking vot2015 challenge

results," in *Proceedings of the IEEE international conference on computer vision workshops*, 2016, pp. 1–23.

[60] H. Lee, Alex, and K. Daijin, "Salient region-based online object tracking," in *IEEE Winter Conference on Applications of Computer Vision*, 2017, pp. 1170–1177.

[61] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *European conference on computer vision*. Springer, 2014, pp. 254–265.

[62] O. Akin, E. Erdem, A. Erdem, and K. Mikolajczyk, "Deformable part-based tracking by coupled global and local correlation filters," *Journal of Visual Communication and Image Representation*, vol. 38, pp. 763–774, 2016.

[63] T. Vojir, J. Noskova, and J. Matas, "Robust scale-adaptive mean-shift for tracking," *Pattern Recognition Letters*, vol. 49, pp. 250–258, 2014.

[64] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4834–4843.

[65] T. Yang and A. B. Chan, "Learning dynamic memory networks for object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 152–167.

[66] A. S. Tripathi, M. Danelljan, L. Van Gool, and R. Timofte, "Tracking the known and the unknown by leveraging semantic information." in *BMVC*, vol. 2, 2019, p. 6.

[67] Z. Zhang and H. Peng, "Ocean: Object-aware anchor-free tracking," *arXiv preprint arXiv:2006.10721*, 2020.

[68] L. Zheng, M. Tang, H. Lu *et al.*, “Learning features with differentiable closed-form solver for tracking,” *arXiv preprint arXiv:1906.10414*, 2019.

[69] M. Mueller, N. Smith, and B. Ghanem, “Context-aware correlation filter tracking,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR)*, 2017, pp. 1396–1404.

[70] M. H. Abdelpakey and M. S. Shehata, “Domainsiam: Domain-aware siamese network for visual object tracking,” in *International Symposium on Visual Computing*. Springer, 2019, pp. 45–58.

[71] M. Danelljan, A. Robinson, F. Shahbaz Khan, and M. Felsberg, “Beyond correlation filters: Learning continuous convolution operators for visual tracking,” in *ECCV*, 2016.

[72] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, “Distractor-aware siamese networks for visual object tracking,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 101–117.

[73] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang, “Learning spatial-temporal regularized correlation filters for visual tracking,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4904–4913.

[74] R. Tao, E. Gavves, and A. W. Smeulders, “Siamese instance search for tracking,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1420–1429.

[75] Y. Zhang, L. Wang, J. Qi, D. Wang, M. Feng, and H. Lu, “Structured siamese network for real-time visual tracking,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 351–366.

[76] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang, "Vital: Visual tracking via adversarial learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8990–8999.

[77] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1763–1771.

[78] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin Zajc, T. Vojir, G. Häger, A. Lukežič, and G. Fernandez, "The visual object tracking vot2016 challenge results," Springer, Oct 2016. [Online]. Available: http://www.springer.com/gp/book/9783319488806

[79] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 749–765.

[80] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration," *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, vol. 6, 2017.

[81] Nvidia, "Nvidia/apex," July 2020. [Online]. Available: https://github.com/NVIDIA/apex

Chapter 8

Conclusion

In this chapter, we conclude the main contribution in this thesis and provide potential extension to the presented work.

8.1 Conclusions

Based on the presented work throughout this thesis, the following conclusions can be drawn:

- A novel framework for object tracking to fuse the predicted output of multiple trackers and select the best prediction to cover more challenging scenarios. The proposed framework has the ability to expand by allowing the user to include more trackers (i.e., user plugins). Moreover, a novel mechanism, called reporter, has been proposed to intervene whenever all trackers fail to track the object of interest. The reporter mechanism is based on particle filter and provides a correction to the trajectory of the object of interest. To accurately calculate the robustness score, a novel metric has been proposed, namely virtual vector.

- A novel end-to-end network tracker has been proposed, the network is designed based on the Densely-Siamese network, called DensSiam. The key idea is to capture the non-local features which are robust to appearance changes. The novelty of this architecture additionally reduces the number of shared parameters in the network while building up a deeper network compared to traditional Siamese network. Furthermore, an effective response map based on self-attention has been proposed to increase the localization performance of the tracker. This response map contains non-local features along with the semantic information about the object of interest. This novel design tackles the vanishing-gradient problem and leverage the feature re-use ability to boost the generalization in the tracker.

- A novel network tracker has been proposed to capture the domain-are features with semantic objectness information, dubbed DomainSiam. The domain-aware features enables the appearance model to be robust to challenging scenarios such as scale changes and illumination variations. This network produces its prediction (i.e., the bound box) based on the most important feature channels. Consequently, this leads to huge reduction to sparsity problem and overhead calculations. To train this novel network, a novel differentiable weighted-dynamic loss function has been proposed specifically for visual object tracking. This loss fucntion is monotonic with respect to its hyper-parameters, leading to higher performance in case of high-dimensional data and non-convexity.

- A novel network design for object tracking, called DP-Siam, has been proposed. DP-Siam consists of a dynamic Siamese Agent-Environment networks that formulate the object tracking in reinforcement learning framework. DP-Siam produces a continuous action that predicts the optimal object location. DP-Siam has a novel architecture that consists of three networks: an Agent network to predict the optimal state of the object being tracked, an Environment network to get the Q-value during the offline training phase to minimize the error of the loss function, and a Siamese network to produce a heat-map. The Environment network acts as a verifier to the action of the Agent network during online tracking. The proposed architecture allows the tracker to dynamically select the hyper-parameters in each frame instead of the traditional method of fixing their values for the entire dataset, which to the best knowledge of the authors, has not been done before.

- Most of trackers work in feature space, meaning that the backbone network is trained on classification task by categorical cross entropy. To design a network that is specifically suitable for classification and tracking, a novel network, namely NullSpaceNet, has been proposed. The network learns to project traditional feature space onto a joint-nullspace. This design ensures that the projected features from the same class are collapsed into a single point while ones from different classes are collapsed into different points with high septation margin. NullSpaceNet is architecture-agnostic that can be applied to any network with fully connected layers. A novel differentiable loss function has been proposed to train the developed NeullSpaceNet. This loss function is totally different from the categorical cross-entropy function. The proposed loss takes into consideration the projection onto the joint-nullspace. Moreover, NullSpaceNet features have a clear interpretation both mathematically and geometrically compared to standard feature space.

- A novel formulation for feature learning in object tracking has been proposed, dubbed NullSpaceRDAR. In nullSpaceRDAR, the feature learning in the backbone network is regularized by projecting onto a joint-nullspace. This formulation ensures that the same target-specific information are collapsed into one point in the learned space while the different target-specific information are collapsed into a different points in the learned space. This leads to high discriminative power to between-target-specific scatter matrix and decrease the within-target-specific scatter matrix. Moreover, to accurately locate the object of interest, a dynamic and adaptive loss function as been proposed in the context of visual tracking to select the best loss function from a super-set family. NullSpaceR-DAR is trained in end-to-end manner and significantly increases the robustness and the generalization of the tracker.

8.2 Future Research Directions

The work presented in this thesis has a number of different future extensions. These directions aim at increasing the object tracking robustness, accuracy, and the expected average overlap.

- Using moving horizon estimation to act as a trajectory recovery mechanism when the tracker fails to locate the object of interest. This mechanism ensures to increase the robustness of the tracker.
- Improving the target estimator speed and accuracy using recently proposed transformers [1] for object detection. As a matter of fact, improving in object detection task leads to improving in object tracking, however, it needs major modifications.
- Incorporating the uncertainty and conditional probability in both target estimator and target classification. Due to the unconstrained environment in object tracking, it is better to estimate the conditional probability score instead of the confidence score such as the work in [2].

- Utilizing more metrics to learn by deep networks. For example, Siamese networks learn a similarity metric, however, it does not use background information, instead, F1-Score can be re-formulated to be differentiable with respect to the weights of the network. This way the network can learn more metrics and including the background information in end-to-end manner.
- Force the network to adaptively predict hyper-parameters during the online tracking using Bayesian inference within reinforcement learning framework.

In summary, future direction can be in two directions: 1) Developing a fast and accurate regression network to regress the coordination of the bounding box and 2) Developing a dedicated classifier for object tracking that is a class agnostic.

8.3 List of Publications

During the Ph.D. period, a number of scientific articles have been published/submitted or are to be submitted. The core chapters of the thesis , i.e., Chapters 2, 3, 4, 5 , 6, and 7, are the publications 1, 7, 8, 2, 3, and 4 of the list, respectively.

Journal Articles:

1. M. H. Abdelpakey, M. S. Shehata, M. M. Mohamed, and M. Gong, “Adaptive framework for robust visual tracking,” IEEE Access, vol. 6,pp. 55273–55283, 2018.

2. M. H. Abdelpakey and M. S. Shehata,“Dp-siam: Dynamic policy siamese network for robust object tracking, ”IEEE Transactions on Image Processing, vol. 29, pp. 1479–1492,

2019.

3. M. H. Abdelpakey and M. S. Shehata,“NullSpaceNet: Nullspace Convoluional Neural Network with Differentiable Loss Function,” IEEE Transactions on Pattern Analysis and Machine Intelligenc (TPAMI), submitted.

4. M. H. Abdelpakey and M. S. Shehata,“NullSpaceRDAR: Regualrized Discriminative Adaptive NullSpace for Robust Visual Object Tracking,” IEEE Transactions on Image Processing, submitted.

   I was the primary author, with author 2 contributing to the idea, its formulation and development, and refinement of the presentation.

5. S. ELbishlawi, M. H. Abdelpakey, A. Eltantawy, M. S. Shehata, and M. M. Mohamed, “Survey On Deep learning-based Crowd Scene Analysis,” MDPI, (accepted)

6. S. ELbishlawi, M. H. Abdelpakey, and M. S. Shehata, “CORONA-Net: COnvolution Re-initialization Optimization Network for Detecting Covid-19 from CT and X-Ray images,” (to be submitted)

   Conference Articles:

7. M. H. Abdelpakey, M. S. Shehata, and M. M. Mohamed, “DensSiam: End-to-end denselysiamese network with self-attention model for object tracking,” in International Symposium on Visual Computing. Springer, 2018, pp. 463–473. (Oral presentation, the method and results in this paper ranked 14th out of 72 papers participated from all over the world in international competition organized and published by VOT community in ECCV2018)

8. M. H. Abdelpakey and M. S. Shehata, “Domainsiam: Domain-aware siamese network for visual object tracking,” in International Symposium on Visual Computing. Springer, 2019, pp. 45–58. (Oral presentation, nominated for the best paper)

9. M. Kristan, A. Leonardis, ..., M. H. Abdelpakey, M. S. Shehata, ..., “The sixth visual object tracking vot2018 challenge results,” Proceedings of the European Conference on Computer Vision (ECCV). pp. 0-0 (2018)

10. M. H. Abdelpakey and M. S. Shehata, “Tracking Objects in Wide Area Motion Imagery (WAMI) using Partial Least Squares Analysis in Noisy Environment,” Proceedings of the Newfoundland Electrical and Computer Engineering Conference (NECEC’16), 2016

11. M. H. Abdelpakey and M. S. Shehata, “Smart Camera Prototype for Automatic Target Recognition,” Proceedings of the Newfoundland Electrical and Computer Engineering Conference (NECEC’15), 2015

References

[1] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” *arXiv preprint arXiv:2005.12872*, 2020.

[2] M. Danelljan, L. V. Gool, and R. Timofte, “Probabilistic regression for visual tracking,”

in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7183–7192.

Appendix A

Loss function of NullSpaceNet

In this appendix, we provide our derivations for the derivative of the loss function. The first part of the appendix is the analytical derivative of the loss function w.r.t the $W^T \tilde{S}_b W$. The second part is the computational graph derivation of the $W^T \tilde{S}_b W$ w.r.t the last layer of NullSpaceNet H as shown in Fig. A.1.

A.1 Analytical Derivative of The Loss Function w.r.t $W^T \tilde{S}_b W$

The nullspace of the within-class scatter matrix is defined as follow:

$$\begin{aligned} U_w &= \left\{ u \in \mathbb{R}^d \mid S_w u = 0 \right\} \\ &= \left\{ u \in \mathbb{R}^d \mid u^\top S_w u = 0 \right\} \\ &= \left\{ u \in \mathbb{R}^d \mid (F_w^\top u)^\top F_w^\top u = 0 \right\} \\ &= \left\{ u \in \mathbb{R}^d \mid F_w^\top u = 0 \right\}. \end{aligned} \tag{A.1}$$

From Equation (20) in the paper, the loss function solves for the Singular Value Decomposition (SVD) as follows:

$$\mathcal{L}(\phi_E(x;\theta)) = -\sum_{i=1}^{k} SVD_{E_i}(W^T \tilde{S}_b W) \tag{A.2}$$

For simplicity let $\mathcal{L} = \mathcal{L}(\phi_E(x;\theta))$, $\frac{\partial \mathcal{L}}{\partial A} = \partial A$, and $A = W^T \tilde{S}_b W$. The SVD of A is given by:

$$A = UEV^T \tag{A.3}$$

Since the loss function adds up the Eigenvalues, we will derive the derivative of the loss function w.r.t three components $U, E,$ and V. Note that the derivative here assumes the low rank case where $A \in \mathbb{R}^{m \times n}$ of rank k and $k \leq \min(m, n)$. The derivative of A w.r.t each component is given by:

$$\partial \mathcal{A} = \partial UEV^T + U\partial EV^T + UE\partial V^T \tag{A.4}$$

From the properties of the SVD and by imposing the orthogonality of U and V, this condition holds:

$$U^TU = V^TV = I \tag{A.5}$$

Where I is the identity matrix. Hence, the derivative of the Equitation A.5 w.r.t U, similarly for V, is given by:

$$\partial U^TU + U^T\partial U = 0 \tag{A.6}$$

Equation A.6 satisfies the fact that the transpose of the matrix eqauls its negative as follows:

$$\partial U^TU = -U^T\partial U \tag{A.7}$$

which means that both sides in Equation A.7 are anti-symmetric matrices. Multiplying the right hand side of Equation A.7 by U and using the condition in Equation A.5 :

$$U \underbrace{U^T \partial U}_{\partial \alpha_u} = \partial U \tag{A.8}$$

From Equation A.8 the derivative of U, similarly for V, is:

$$\partial U = U \partial \alpha_u \tag{A.9}$$

Since U is orthogonal:

$$\partial \alpha_u = U^T \partial U \tag{A.10}$$

In our implementation we do not fully use all basis vectors however, in the back-propagation all basis have an effect on the accumulated error. Hence, using Gram-Schmidt we add the orthogonal complement $U^{\perp} \partial \beta_u$ to ∂U as follows

$$\partial U = U \partial \alpha_u + U^{\perp} \partial \beta_u \tag{A.11}$$

Multiplying both sides of Equation A.4 by $U^T V$

$$\begin{aligned} U^T \partial A V = \underbrace{U^T \partial U}_{\partial \alpha_u} E \underbrace{V^T V}_{I} + \\ \underbrace{U^T U}_{I} \partial E \underbrace{V^T V}_{I} + \\ \underbrace{U^T U}_{I} E \partial \underbrace{V^T V}_{\partial {\alpha_v}^T} \end{aligned} \tag{A.12}$$

The equation A.12 becomes:

$$U^T \partial A V = \partial \alpha_u E + \partial E + E \partial {\alpha_v}^T \tag{A.13}$$

In Equation A.13, since the $\partial\alpha_u$ and $\partial{\alpha_v}^T$ are anti-symmetric matrices, the right diagonals are zero or almost zero. This property, anti-symmetric, forces the Equation A.13 to be split into two equations, the on-diagonal component and off-diagonal component:

$$I \cdot U^T \partial A V = \partial E \tag{A.14}$$

$$\tilde{I} \cdot U^T \partial A V = \partial\alpha_u E + E \partial{\alpha_v}^T \tag{A.15}$$

$\tilde{I}$ is the complement of I where the right diagonal is zeros and all other elements are non-zero. Using the property of anti-symmetric:

$$\tilde{I} \cdot U^T \partial A V = \partial\alpha_u E - E \partial\alpha_v \tag{A.16}$$

Taking the transpose of Equation A.16

$$\tilde{I} \cdot (U^T \partial A V)^T = -E \partial\alpha_u + \partial\alpha_v E \tag{A.17}$$

Multiplying both Equations A.16 and A.17 by $E$and adding up:

$$\begin{aligned} \tilde{I} \cdot [U^T \partial A V E + E (U^T \partial A V)^T] = \\ \partial\alpha_u E^2 - E^2 \partial\alpha_v \end{aligned} \tag{A.18}$$

Rearrange Equation A.18

$$\partial\alpha_u = F \cdot [U^T \partial A V E + E (U^T \partial A V)^T] \tag{A.19}$$

Where F is defined as follows

$$F_{ij} = \begin{cases} \frac{1}{E_j^2 - E_i^2} & i \neq j \\ 0 & i = j \end{cases} \tag{A.20}$$

Similarly for $\partial\alpha_v$

$$\partial\alpha_V = F \cdot [EU^T\partial AV + (U^T\partial AV)^T E] \tag{A.21}$$

To find $\partial\beta_u$, multiply Equation A.4 by $\perp U^T$ and substituting by Equation A.11

$$\begin{aligned} \perp U^T\partial\alpha_V =& \perp U^T\partial UEV^T + \\ &\underbrace{\perp U^TU\partial EV^T + \perp U^TUE\partial V^T}_{0} \\ =& \perp U^T\partial UEV^T \\ =& \underbrace{\perp U^T[U\alpha_u}_{0} + U^{\perp}\partial\beta_u]EV^T \\ =& \partial\beta_u EV^T \end{aligned} \tag{A.22}$$

Consequently, $\partial\beta_u$, similarly for $\partial\beta_v$, can be re-written as follows:

$$\partial\beta_u = U^{\perp}\partial AVE^{-1} \tag{A.23}$$

Finally, Using the property of anti-symmetric, the derivative of U is given by:

$$\begin{aligned} \partial U = U[F \cdot [U^T\partial AVE+ \\ EV^T\partial A^TU]]+ \\ [I - UU^T]\partial AVE^{-1} \end{aligned} \tag{A.24}$$

∂E is given below:

$$\partial E = I \cdot U^T\partial AV \tag{A.25}$$

∂V as follows:

$$\begin{aligned} \partial V = V[F \cdot [EU^T\partial AV+ \\ V^T\partial A^TUE]]+ \\ [I - VV^T]\partial A^TUE^{-1} \end{aligned} \tag{A.26}$$

A.2 Computational Graph Derivation of $W^T\tilde{S}_bW$

The computational graph of the proposed loss function is shown in Fig. A.1. From Equation 11 in this chapter let these variables are defined as follows:

$$\mu_h = \mu_g, \tilde{H} = F_t, \tilde{H}_c = F_w \tag{A.27}$$

Assuming the forward pass has been executed once and all needed variables are cached, using the chain rule we can derive the derivative of $W^T\tilde{S}_bW$ w.r.t the last layer of NullSpaceNet as shown in Fig. A.1 as follows:

$$\partial\tilde{S}_b = U\partial EV^T(sqw) \tag{A.28}$$

$$\partial(sqw) = \sum U\partial EV^T\tilde{S}_b \tag{A.29}$$

$$\partial(squ_1) = \partial\tilde{S}_bS_b \tag{A.30}$$

$$\partial S_b = \partial\tilde{S}_b(squ_1) \tag{A.31}$$

$$\partial W = 2W\partial(sqw) \tag{A.32}$$

$$\partial(\tilde{S_{w_2}}) = \partial V \partial W \tag{A.33}$$

$$\partial(squ_2) = \partial(\tilde{S_{w_2}}) S_w \tag{A.34}$$

$$\partial S_{w_2} = (squ_2) \partial \tilde{S_w} \tag{A.35}$$

$$\partial u_1 = 2[\partial(squ_1) + \partial(squ_2)] \partial(\tilde{S_{w_2}}) u_1 \tag{A.36}$$

$$\partial \tilde{H}_1 = \partial u_1 \partial A \tag{A.37}$$

$$\partial S_t = (\partial S_b) S_w \tag{A.38}$$

$$\partial S_{w_1} = -\partial S_b \tag{A.39}$$

$$\partial sq_1 = \frac{1}{N} I_{(N,D)} \partial S_t \tag{A.40}$$

$$\partial \tilde{H}_2 = 2\partial(sq_1)\tilde{H} \tag{A.41}$$

$$\partial H_1 = \partial \tilde{H}_1 + \partial \tilde{H}_2 \tag{A.42}$$

$$\partial H_{\mu_h} = -[\partial \tilde{H}_1 + \partial \tilde{H}_2] \tag{A.43}$$

$$\partial(sq_2) = \frac{1}{N} I_{(N,D)}[\partial S_{w_1} + \partial S_{w_2}] \tag{A.44}$$

$$\partial \tilde{H}_c = 2\tilde{H}_c \partial(sq_2) \tag{A.45}$$

$$\partial H_2 = \partial \tilde{H}_c \tag{A.46}$$

$$\partial \mu_c = -\sum \partial \tilde{H}_c \tag{A.47}$$

$$\partial H_3 = \frac{1}{C} \partial \mu_c I_{(N,D)} \tag{A.48}$$

$$\partial H_4 = \frac{1}{m} I_{(N,D)} \partial \mu_h \tag{A.49}$$

Finally, $\frac{\partial \mathcal{L}(\phi_E(x;\theta))}{\partial H}$ is calculated as follows

$$\frac{\partial \mathcal{L}(\phi_E(x;\theta))}{\partial H} = \partial H_1 + \partial H_2 + \partial H_3 + \partial H_4 \tag{A.50}$$

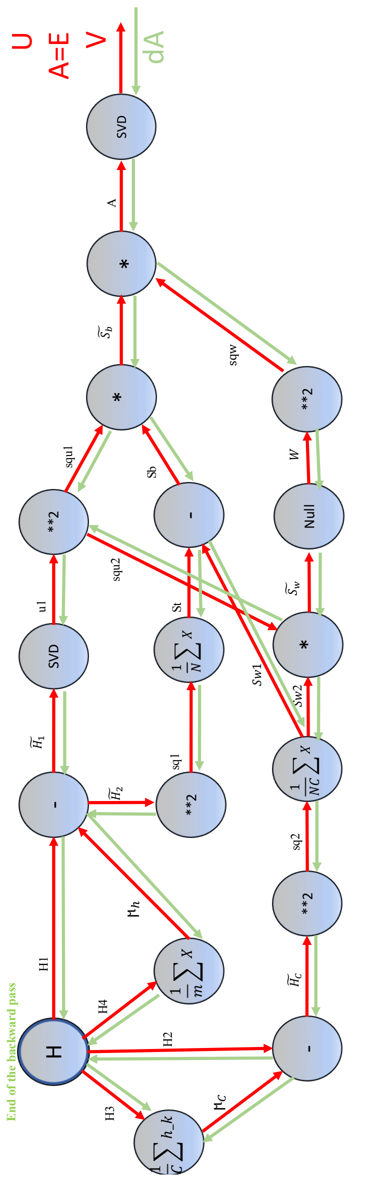


Figure A.1: Computational graph of the proposed loss function.The red arrows are the forward pass and the green arrows are the backward pass. Each arrow has a variable name on it.