Visual Object Tracking in Dynamic Scenes

A

Thesis submitted

for the award of the degree of

DOCTOR OF PHILOSOPHY

By

Mathew Francis

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, ASSAM, INDIA

September 2023

Certificate

This is to certify that the thesis entitled "**Visual Object Tracking in Dynamic Scenes**", submitted by **Mathew Francis** (146102001), a research scholar in the *Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati*, for the award of the degree of **Doctor of Philosophy**, is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute and in my opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dated: 30-September-2023

Dr. Prithwijit Guha
Associate Professor
Dept. of Electronics and Electrical Engg.
Indian Institute of Technology Guwahati
Guwahati - 781039, Assam, India.

To

My sweet and loving family,

for their love, affection, encouragement and prayers

Acknowledgements

This thesis would not have been possible without the immense help and support of several people in various measures. I take this opportunity to convey my most sincere acknowledgement to all of them. I would like to express my deepest and most heartfelt gratitude to my supervisor Dr. Prithwijit Guha, for the constant guidance, help and encouragement throughout my research work. His insightful feedbacks have helped me much in improving the quality of my thesis.

I am grateful to Prof. P.K. Bora, the Chairman of the Doctoral Committee for providing valuable suggestions on my work throughout the years. I want to express my sincere gratitude to Dr. Tony Jacob and Dr. Arijit Sur, distinguished members of the Doctoral Committee for their constructive criticisms during my progress seminars.

I would also like to convey my gratitude to all technical and non-technical staffs of the EEE department for their help and support throughout my Ph.D. duration; especially, I acknowledge Mukut for timely forwarding of different applications.

I take this opportunity to extend my gratitude towards my senior, Dr. Raghvendra Kannao, Dr. Sandeep P. for various technical discussions and suggestions they provided whenever I needed. Useful technical discussions with them shaped my research in many aspects. A thankful note to past members of the lab Dr. Shikha Baghel, Dr. Mrinmoy Bhattacharjee.

The stay at the IITG campus would not have been comfortable without the facilities provided by the institute. The campus life remained lively in the presence of my friends, Dr. Sishir Kalita, Dr. Kaushik, Dr. Vivek Venugopal, Dr. Alex Paul Kamson, Dr. Vijith Kumar, Rahul R. Nair. Their constant help and support are invaluable during my life in IITG.

I am incredibly fortunate to have a caring and loving family who deserve special mention for providing mental strength and encouragement to explore my potentials and pursue my dreams. With great pleasure, I would like to give special thanks to my wife, Betsy. Finally, I thank God Almighty for the blessings he has bestowed upon me and for giving me the wisdom to achieve this dream.

Mathew Francis

Abstract

Visual object tracking is one of the popular and legacy problems in computer vision. Tracking has applications in a wide spectrum of domains with applications in agriculture, automobile, surveillance, defence, and entertainment. It remains a challenging research problem on account of various factors like occlusions, shape deformations, illumination changes and background clutter. The object tracking performance depends on the joint efficiencies of a target object modeling scheme and a localization strategy. Accordingly, this thesis proposes three major contributions. First, a colour feature based representation that models the foreground while considering the background context. The target is further localized by using a meta-heuristic search strategy based on breeding fireflies. The breeding fireflies technique is realized through a combination of Real Coded Genetic Algorithm (RGA) and the Firefly Algorithm (FA). The second contribution involves target representation in the sparse representation framework. Here, the object is modeled using a (weighted) distribution of sparse codes. The weights are derived from a foreground-background classifier. Here, the target is localized by using the Firefly-RGA approach. The third contribution uses target representation using deep network embeddings. Here, a Siamese network is used to derive the object location predictions. Additionally, a multi-part object model is explored for handling occlusions. All three proposals use the appearance features to predict the object positions. Thus, complementary information is derived by inter-frame dense motion estimation. The motion based predictions are used to enhance the accuracy of the three proposed appearance based trackers. Thus, all the proposed trackers are realized in an ensemble framework of appearance and motion based predictions. The colour feature based tracker has an object model based on handcrafted features derived from colour distribution. The sparse representation-based tracker learns the model by using features from only the first frame of the sequence. In the Siamese tracker, a pre-trained deep network is used for extracting image features. The performances of the classical methods are limited by the feature used. For example, colour distribution-based features get disturbed by illumination

changes and structure-based features face challenges from geometric transformations. In contrast, the deep network used in the Siamese tracker is trained on a very large set of general images (for image classification task) and is thus capable of generating rich feature representations. It is also observed that the Siamese tracker has significantly outperformed the other two trackers. The proposed trackers are benchmarked on the $VOT2018$, $OTB2015$ and $UAV123$ datasets and are compared against a number of baseline methods formulated in traditional and deep learning framework.

Keywords: tracking, modeling, localization, sparse, convolution, siamese, motion

Contents

List of Figures

List of Tables

List of Acronyms

AUC	Area Under the Curve
CNN	Convolutional Neural Network
DE	Differential Evolution
EAO	Expected Average Overlap
GA	Genetic Algorithm
GOT	Generic Object Tracking
IOU	Intersection over Union
LSTM	Long Short Term Memory
MOT	Multiple Object Tracking
OTB	Object Tracking Benchmark
PSO	Particle Swarm Optimization
SOT	Single Object Tracking
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SNT	Siamese Network based Tracker
RNN	Recurrent Neural Network
VOT	Visual Object Tracking

List of Symbols

a	Annotation	$\mathcal{T}$	Tracker
E	Embedding	$\boldsymbol{\mathcal{I}}^S$	Search Image
$\boldsymbol{\mathcal{I}}$	Image	$\boldsymbol{\mathcal{I}}^Q$	Query Image
$\mathbf{bb}$	Bounding Box	Φ	Network
$\mathbf{\Gamma}^+$	Sparse Code Matrix of Object Patches	$\boldsymbol{\gamma}^-$	Sparse Code Vector of Background Patch
$\boldsymbol{\gamma}^+$	Sparse Code Vector of Object Patch	$\mathbf{\Gamma}^-$	Sparse Code Matrix of Background Patches
θ	Rotation	$\mathbf{D}^+$	Foreground Dictionary
ρ	Bhattacharyya Coefficient	$\mathbf{D}^-$	Background Dictionary
$g()$	Kernel	$\mathbf{D}$	Dictionary
L	Normalization Constant	n_K	Cluster count
$\mathbf{m}^{\mathbb{T}+}$	Target Foreground Model	$\mathcal{L}$	Loss
$\mathbf{m}^{\mathbb{T}-}$	Target Background Model	η_s	Sparsity
$\mathbf{m}^{\mathbb{C}+}$	Candidate Foreground Model	$\boldsymbol{\mathcal{I}}^{\mathbb{C}}$	Candidate Patch Image
$\mathbf{m}^{\mathbb{C}-}$	Candidate Background Model	$\mathbb{J}$	Objective Function
$\mathbb{X}$	Patch Vectors Matrix	Ω	Search Space
$\mathbb{P}$	Population	$\mathbb{X}^-$	Patch Matrix of Background Patches
$\mathbb{X}^+$	Patch Matrix of Object Patches	τ	Iteration
$\mathcal{S}$	Sequence	$\overset{\star}{\mathbf{bb}}^+$	Predicted bounding box
ρ	Similarity Measure	$\mathbb{B}^+$	Foreground bounding box set
$\mathbf{z}$	State Vector	$\mathcal{O}^F$	Optical flow
Ψ	Performance Measure	$\mathbb{S}$	Scoremap

1

Introduction

Contents

Vision is one of the primary senses of human beings. We are extremely skilled at performing visual tasks. For example, detection, recognition and tracking of specific objects of interest in a scene is a simple manual task. However, something that we excel at, could be quite challenging for a machine to accomplish. Visual object tracking is one among such tasks.

Vision based *single object tracking* refers to the process of following an object of interest in a sequence of images. An illustration of tracking performed on the butterfly sequence from $VOT2018$ dataset is shown in Figure 1.1. The target object is initalized in the first frame and is localized in the remaining frames of the sequence. The target object localization and tracking across video frames have several applications in the areas of entertainment, defence, healthcare, and security-surveillance [1]. Some of these are shown in Figure 1.2.

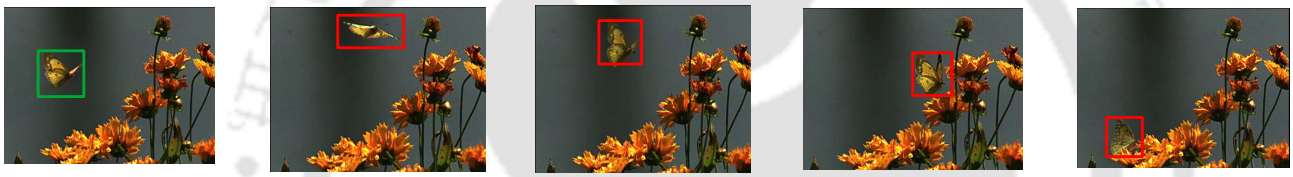

Figure 1.1: Object to be tracked is initialized in first frame(Green box). Object is localized in the remaining frames by the tracking algorithm (Red box)

Figure 1.2: Some applications of visual object tracking like traffic analysis, green screen marker tracking, crowd analysis, autonomous vehicle and surveillance.

This thesis focuses on generic single-object tracking with RGB data in a dynamic single camera configuration with variable execution speed. Figure 1.3 shows the broad categorization of approaches based on the factors external to a tracking algorithm. This includes the type and number of imaging sensors, availability of data, number of objects being tracked, speed of operation and the type of object being tracked. The object to be tracked is generic, implies that the tracker is not limited to finding only a specific class of object. No information about the object is available prior to tracking. Data is made available to the algorithm on a streaming manner making sure that future frames are not accessible to the algorithm at any time instant. First frames of the sequence contains information on the object and its surroundings. Other frames are initially unavailable when tracking starts. If neces-

sary, tracking models can be learned by using all frames of the videos. However, this thesis does not adopt such a technique. The colour-based tracker learns the model from the first frame (initialization phase) and occasionally updates it in the localization (operational) phase. The sparse tracker uses the first frame only to learn the model in the initialization phase. Similarly, the deep tracker also uses the object template obtained from the first frame for tracker initialization. No updates are performed in the operation phase. Also, only pre-trained deep network is used for image feature extraction for the Siamese tracker. Thus, the proposed methods (especially, the sparse and Siamese trackers) do not employ any “model learning" in the localization phase.. Color, temperature, depth, motion, and other aspects of a scene can be captured via sensors. Only colour data, generally known as RGB data, is dealt with in algorithm development. Single or multiple cameras can be used in a tracking system to cover the same or multiple scenes. In this thesis, a single camera configuration is assumed that can be either stationary or moving.

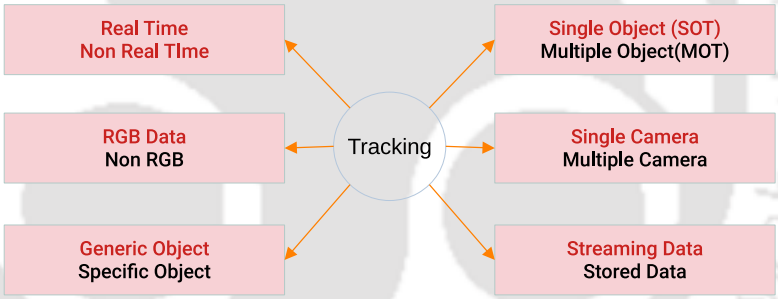

Figure 1.3: Categories of tracking. Sub-categories of interest in this work are colored red

1.1 Problem Formulation

Let $\mathcal{S}$ be a sequence of n^T frames defined as $\mathcal{S} = \{\boldsymbol{\mathcal{I}}_t, t = 0, \ldots, n^T\}$. Here, $\boldsymbol{\mathcal{I}}_t$ represents the t^{th} frame of the sequence. The target object is identified in the initial frame $\boldsymbol{\mathcal{I}}_0$ with its annotation $a(\boldsymbol{\mathcal{I}}_0)$. The goal of the tracker $\mathcal{T}$ is to predict object bounding regions on frames $\{\boldsymbol{\mathcal{I}}_t, t = 1, \ldots, n^T\}$ of the sequence given $a(\boldsymbol{\mathcal{I}}_0)$ and $\boldsymbol{\mathcal{I}}_t$. Let the tracker prediction on frame $\boldsymbol{\mathcal{I}}_t$ be represented as $\mathcal{T}(\boldsymbol{\mathcal{I}}_t)$. This is shown as in Figure 1.5.

Despite its popularity and extensive research, tracking remains a challenging computer vision problem. The photometric, geometric, and environmental variations are some of the most common challenges in tracking. Geometric changes happen with view point changes and camera displacements or zooming. Here, an object can undergo shape deformations, translation, scaling, orientation changes

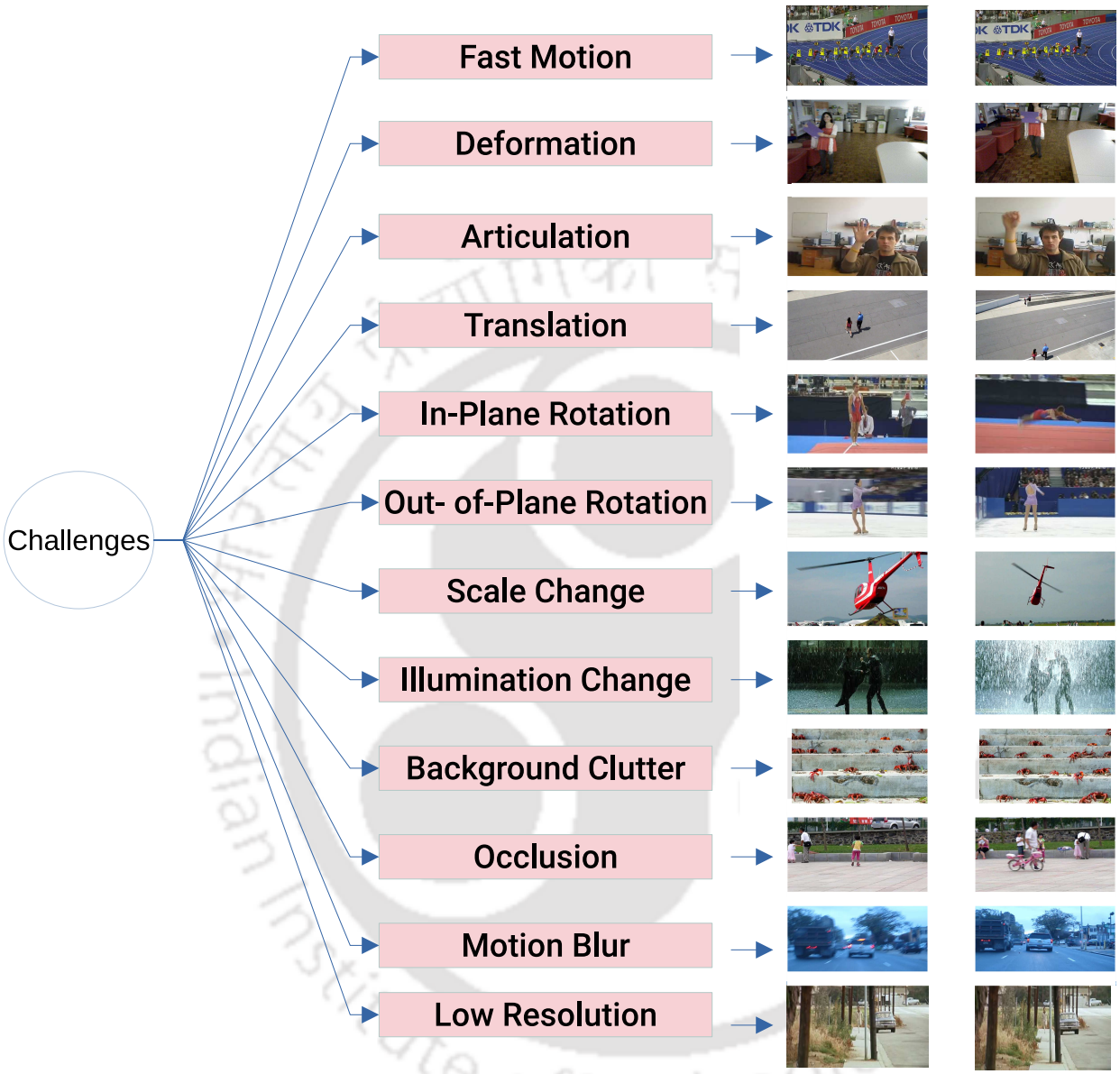

Figure 1.4: The different challenging scenarios of visual object tracking.

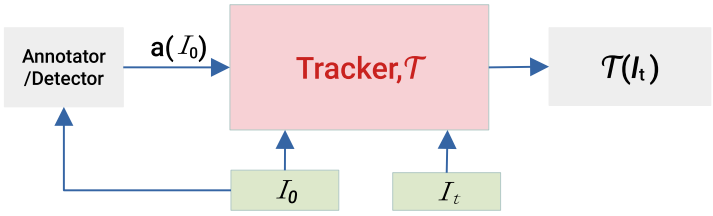

Figure 1.5: Tracking problem definition.

and articulations. Radiometric challenges include illumination changes and shadows. These factors directly affect the target object appearance. External factors like background clutter and occlusion also adversely affect the tracking performance. The tracker may gradually drift from the target on account of the above mentioned challenging scenarios. However, such drift also occurs due to improper object modeling or model update schemes. This leads to localization errors and progressive loss of the trail of target. A tracker's success depends on its ability to overcome these conditions. Some of the common challenges in tracking are shown in Figure 1.4.

A tracker has two important components [2]. First, a methodology for target object modeling. The target model is constructed by using hand-crafted or learned features extracted from the object support in image. The model is also occasionally updated in the course of tracking. Ideally, the target model should be robust to the photometric or geometric changes. Also, it should be constructed using features that can discriminate the target from the background. The second component is a strategy for localizing the target in the present frame by using the learned object model. A good localization strategy overcomes occlusions and background clutter. The tracker performance depends on the joint efficiencies of the target model and the localization strategy.

This thesis explores different schemes for target model representation. Target models constructed using both engineered (color distribution and sparse codes) and learned features (deep network embeddings) are explored. The objects are localized using meta-heuristic search strategies and particle filter based approaches. The predictions provided by the object appearance features are further corrected by motion predictions. This leads to the proposal of trackers in an ensemble framework that uses complimentary information. The contributions of this thesis are briefly discussed in the next section.

1.2 Contributions

The existing works in object tracking can be broadly divided into two groups – the classical approaches and the methods formulated in deep learning framework [3]. The classical approaches use hand-crafted features, statistical models and machine learning techniques for object tracking. In this thesis, two methodologies are proposed along the lines of classical approaches. Approaches formulated in deep learning framework follow a paradigm of learning from data with deep neural networks. In this direction, this thesis proposes a siamese network based tracking algorithm.

Object modelling and target localization are the two key components of any tracking algorithm. Additionally, ensemble of trackers are also explored in the literature. This thesis explores generative, discriminative, and hybrid techniques for target modelling. Novel objective functions based on the proposed target and background models are proposed to aid in localization step. Object localization is explored using meta-heuristic search strategies. Finally, inter-frame dense motion is used as a complementary information with respect to appearance based predictions. Predictions from both appearance and motion are used in an ensemble framework to improve the tracker performance.

The first contribution involves the proposal of a color distribution based object model, corresponding target localization objective function and a novel localization strategy in breeding-swarm framework(Subsection 3.3.4). The second contribution proposes object modeling in sparse representation framework. The object models and corresponding localization objective functions are described in Sections 4.4, 4.5 and 4.6. The third contribution proposes a multiple part based object tracker in the framework of siamese networks (Sections 5.4 and 5.5). Motion information based refinement of appearance feature based tracker predictions are explored for all three tracker proposals.

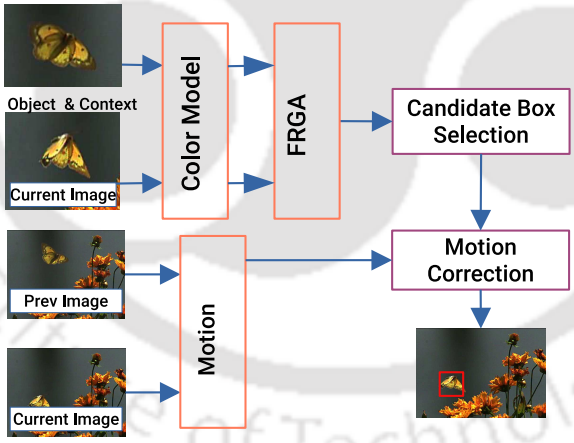

Figure 1.6: Block diagram of color distribution based tracking algorithm. FRGA refers to Firefly algorithm modified with RGA.

1.2.1 Color Distribution based tracker

Chapter 3 proposes a color distribution based object model. Color distribution has been widely used in the tracking literature [4]. Background pixels from the neighborhood of the object are used to boost (target-background) discrimination. It is used to refine the object model by giving emphasis to object specific characteristics. An objective function is proposed for target localization to a region with

maximal object presence and minimal background presence. Both probabilistic and meta-heuristic approaches are used for target localization. A block diagram of color feature based tracker is shown in Figure 1.6. Different localization strategies, such as particle filter [5], particle swarm optimization [6], genetic algorithm [7], and firefly algorithm [8] are investigated.

With these trials, the effect of background information on tracking is investigated. A proposal is made to include solution breeding with Real-Coded Genetic algorithm to firefly population. New solutions are introduced to improve the diversification of a solution set resulting in better search. This approach of breeding firefly population with Real-coded Genetic Algorithm is named FRGA. Further, the predictions from inter-frame dense motion and the color distribution based tracker is formulated in an ensemble framework. The motion component is used to refine the predictions provided by the proposed FRGA algorithm. The proposed tracker is evaluated on three standard datasets – $VOT2018$, $OTB2015$ and $UAV123$. The major contributions of Chapter 3 can be briefly summarized as follows.

(i) A novel Context Weighted Color Distribution (CWCD) based object model

(ii) Novel objective function that includes contextual information

(iii) Modified Firefly algorithm with RGA component

(iv) Motion based re-ranking in ensemble tracker

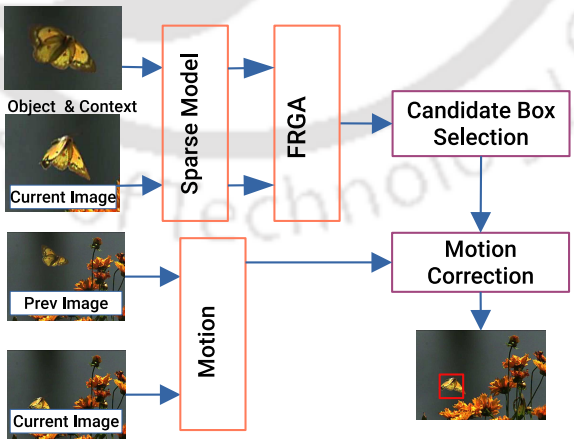

Figure 1.7: Block diagram of sparse feature based tracking algorithm.

The color computed from the entire target object region lacks local structural information. Object modeling in sparse representation framework is explored to represent local information. This is discussed next.

1.2.2 Sparse representation feature based tracker

Chapter 4 explores the sparse representation framework for target model representation. This approach was made popular by the work of [9]. It was developed on a dictionary defined on object template and its perturbations. Such a representation was shown to have robustness against occlusion and noise. This thesis proposes to learn a dictionary using vectorized patches from target object and background region. The learned dictionary is used to compute sparse codes for vectorized patches. A histogram of sparse codes is used to construct the object model. A SVM classifier is also used to improve the discrimination against background.

Two target representations were proposed. The first one in generative, and the second one in hybrid framework. Sparse representations of vectorized image patches were computed after clustering and dictionary learning. Clustering identifies the unique parts of object and background. Learned cluster specific dictionaries are suited for representing each part. Thus, the overall dictionary has atoms from both target and background. Sparse histograms were computed to define the object model in the generative approach. A novel objective function is proposed for localization. The objective function measures the target-similarity of a candidate region while considering its background. A block diagram of sparse representation based tracker is shown in Figure 1.7.

In the hybrid model, a support vector machine based classifier was learned on sparse codes to predict a target similarity score for each patch. Weighted histogram of sparse codes were computed as a representative model. Such a model is learned on different predefined regions of the object. A collection of such weighted histograms computed from different regions forms the overall model. The objective function for target localization is proposed as the Bhattacharya coefficient between target and candidate models. A FRGA based search is used for target localization. Object position, scale and orientation are predicted by the proposed tracker. The motion component is used to refine the predictions provided by the proposed FRGA algorithm. The proposed trackers are evaluated on three standard datasets – $VOT2018$, $OTB2015$ and $UAV123$. The major contributions of Chapter 4 are as follows.

(i) A generative model based on Distribution of Sparse Codes (DSC).

(ii) A hybrid target model built on binary Classifier weighted Distribution of Sparse Codes of object parts (CWDSC).

(iii) Novel objective functions based on the models.

(iv) Sparse representation based trackers with *FRGA* and motion information based re-ranking for localization.

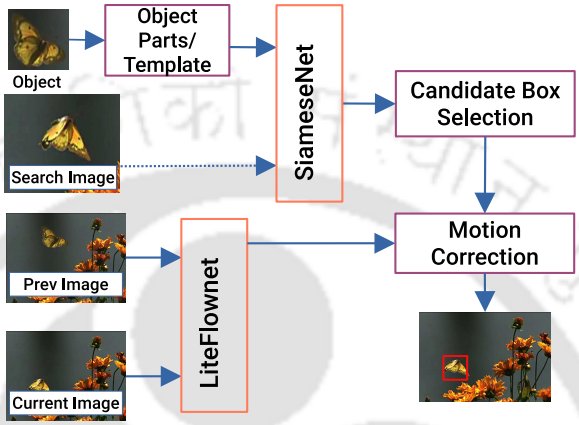

Figure 1.8: Block diagram of siamese network based tracking algorithm.

1.2.3 Siamese network based tracker

Chapter 5 proposes a deep neural network based tracker in an ensemble framework. Here, the object model is defined using the embeddings generated by pre-trained deep neural networks. Convolutional Neural Network (CNN) based representations are known to provide better object representations compared to the hand-crafted ones. This work uses learned features obtained as embeddings from deep networks. Tracking using siamese network was introduced in [10] and further modifications were proposed in later works [11–13]. The siamese network based approach formulates tracking as a similarity search problem. The object appearance embeddings are generated using the pretrained deep networks *VGG-16* [14]. The embeddings of the target object is searched in a given image. The object is localized on the region with highest similarity. Two variations of this approach are proposed. The first proposal uses siamese architecture on the entire object region. The second one divides the object region into predefined parts and applies the siamese tracker on each of them independently. Such part-based object models have been widely used for handling occlusions [15–17]. A block diagram of deep embedding based tracker is shown in Figure 1.8.

In case of the motion component, a dense flow is predicted by using the pretrained deep network *LiteFlowNet* [18]. Generally, the target object and the background have different motion signatures.

This assumption motivated the use of inter-frame dense motion based prediction for target localization. The predictions of the appearance feature based siamese tracker are refined with the motion information. The appearance component predicts potential image regions with confidence values indicating target similarity scores. The motion prediction is used further to estimate the position of every object pixel. The final object position is derived from the agreement between these two predictions. The proposed trackers are evaluated on four standard datasets – $VOT2018$, $VOT2019$, $OTB2015$ and $UAV123$. The major contributions of Chapter 5 can be briefly summarized as follows.

(i) Multi-part object model for handling occlusions

(ii) Motion information based re-ranking for refining the appearance component prediction

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 reviews the existing (and relevant) works from the tracking literature, introduces the standard datasets, and performance evaluation metrics. The color distribution based object model with FRGA based localization are described in Chapter 3. Chapter 4 presents the tracker proposal formulated in sparse representation framework. The multiple part based tracker in the framework of siamese networks is described in Chapter 5. Inter-frame dense motion information based refinement of color (Chapter 3), sparse (Chapter 4) and deep embedding (Chapter 5) based tracker predictions are investigated for all three tracker proposals. Finally, Chapter 6 summarizes the tracker proposals and discusses possible future extensions of the present work.

2

Literature Review

Contents

Overview

Object tracking is one of the widely researched area in computer vision. There are different approaches that are proposed in the literature to handle different challenging scenarios in tracking. The existing works in object tracking can be broadly divided into two groups. First, the classical approaches, and second, the more recent deep learning based methods. The former group of approaches uses engineered features, statistical models and classical machine learning algorithms for tacking. While the later employs different deep neural network architectures for the task of tracking. This chapter briefly reviews the classical and deep learning based approaches. The tracking algorithms are often benchmarked on a few standard datasets. These datasets and the different metrics used for comparative tracking performance evaluation are also discussed.

2.1 Introduction

Object tracking is a widely explored field in computer vision. The existing works in visual tracking can be broadly divided into two groups namely, classical approaches and the ones formulated in deep learning framework. The classical approaches focus on hand-crafted features, statistical and/or machine learning based object model construction and localization strategies [19]. On the other hand, deep neural network based approaches mostly rely on designing architectures for trackers based on end-to-end learnable networks. The contemporary literature is dominated by deep learning based approaches where contributions are made in the proposals of network architecture, learning algorithm, loss functions among others [20]. A tracker has two phases of operation – initalization and target localization.

Initialization Phase – The initialization happens in the first frame $\boldsymbol{\mathcal{I}}_0$, of a sequence $\{\boldsymbol{\mathcal{I}}_0, \ldots, \boldsymbol{\mathcal{I}}_{n^T}\}$. The target object is identified by its annotation on first frame $(a(\boldsymbol{\mathcal{I}}_0))$. Most algorithms use a rectangular region crop of the object from $\boldsymbol{\mathcal{I}}_0$ as the template image $\boldsymbol{\mathcal{I}}^Q$ of the object. A bounding rectangle annotation can be parameterised as follows.

$$\mathbf{bb}_0^+ = [x_0, y_0, w_0, h_0, \theta_0]^T \tag{2.1}$$

Where, x_0, y_0, w_0, h_0, and θ_0 are the respective centroid abscissa , centroid ordinate, width, height and orientation of the bounding rectangle. The feature extraction (from $\mathbf{bb}_0^+$) and target object modeling are performed in this stage. For DNN based approaches, the feature extraction and object modeling

might not be explicit as these are often defined by deep networks.

Figure 2.1: Target foreground and background region defined on frame $\mathcal{I}_0$.

In addition to object information, many algorithms utilize the background information in tracking to improve discrimination. Usually a bigger rectangle with same centroid coordinates (x_0, y_0) and a scaled area of $\mathbf{bb}_0^+$ is defined to include both object and its immediate neighborhood (context).

$$\mathbf{bb}_0^{\pm} = [x_0, y_0, w_0^{\pm}, h_0^{\pm}, \theta_0]^T \tag{2.2}$$

A region consisting only background information is further defined as follows.

$$\mathbf{bb}_0^{-} = \mathbf{bb}_0^{\pm} - \mathbf{bb}_0^{+} \tag{2.3}$$

The regions $\mathbf{bb}_0^+, \mathbf{bb}_0^-$ are referred as target foreground and background regions respectively (Figure 2.1). The image crop defined with $\mathcal{I}_0$ and $\mathbf{bb}_0^-$ is represented as $\mathcal{I}^{Q-}$ which contains only background information.

Feature extraction and object modeling are the two important steps in the intialization stage. Features include pixel intensities, deep feature embeddings among others. For modeling an object statistical tools as well as machine learning and deep learning algorithms are commonly used. Tracking phase starts after initialization phase and exists from frames $\mathcal{I}_1$ to $\mathcal{I}_{n^T}$.

Target Localization Phase – In this phase, the target object is localized in the given frame $\mathcal{I}_t$ to obtain the tracker prediction $\mathcal{T}(\mathcal{I}_t)$. Algorithms can perform localization in the position (x, y), scale (w, h) and orientation θ. This will be in the form of an oriented bounding box $\overset{\star}{\mathbf{bb}}_t^+$. Not all algo-

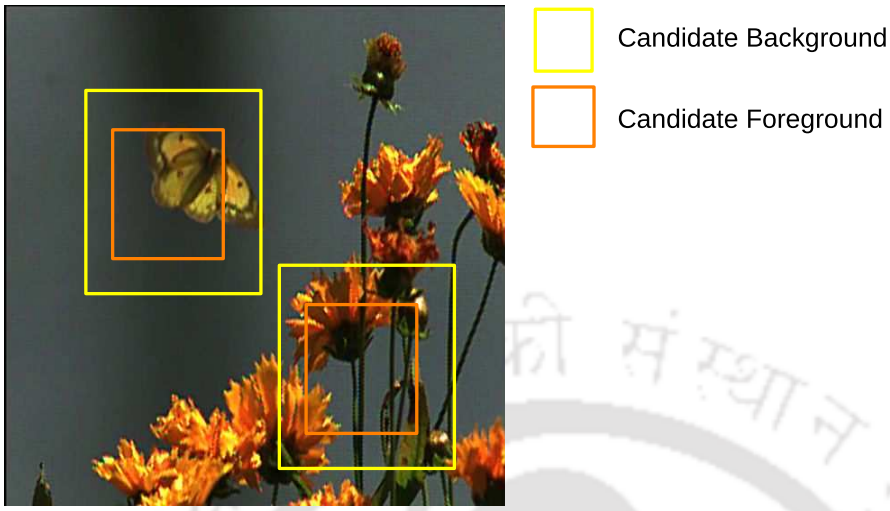

Figure 2.2: Candidate foreground and background region defined on frame $\mathcal{I}_t$.

rithms perform localization in the form of a bounding box. Sometimes only the position is required. Some algorithms go one step further and estimate object contour [21]. Localization algorithms are executed either on the entire image $\mathcal{I}_t$ or on a smaller region defined around the prediction from previous frame ($\overset{\star}{\mathbf{bb}}^{+}_{t-1}$). This small region is referred as the search image $\mathcal{I}^S \in \mathcal{I}_t$.

The search can be performed through brute force techniques [22], iterative methods [4], sampling based probabilistic approaches [5, 21, 23] or meta-heuristic strategies [24–26]. Multiple candidate bounding rectangle proposals $\mathbf{bb}_{ti}$ are generated in the current frame $\mathcal{I}_t$. The set of all such bounding rectangle proposals is defined as

$$\mathbb{B}_t = \{\mathbf{bb}_{ti}, i = 1, \ldots, n_{\mathbb{B}}\} \tag{2.4}$$

where, $n_{\mathbb{B}}$ is the total number of bounding rectangle proposals. Models computed from these candidate bounding rectangles ($\mathbf{bb}_{ti}$) are referred to as candidate models. An objective function is generally defined as similarity (or dissimilarity) between the target and candidate models. A suitable localization approach is adopted to identify the optimal solution $\overset{\star}{\mathbf{bb}}^{+}_{t}$ (from $\mathbb{B}_t$) that optimizes the objective function.

The visual tracking literature is rich with algorithms covering each and every aspect of the two phases [20] i.e. initialization and localization. The later is also known as the operational phase of the tracker. For developing and evaluating an algorithm it is essential to have relevant annotated datasets, evaluations paradigms and performance measures. Standard datasets (available in public

domain), common performance evaluation measures and evaluation paradigms are also presented in this chapter. Rest of the chapter is organised as follows. Section 2.2 discusses the classical approaches in tracking. Section 2.3 reviews the deep learning based approaches. Section 2.4 introduces the popular datasets and performance measures used in the literature. Section 2.5 discusses the evaluation strategies. The review is summarized in Section 2.8 while identifying the research directions.

2.2 Classical Approaches

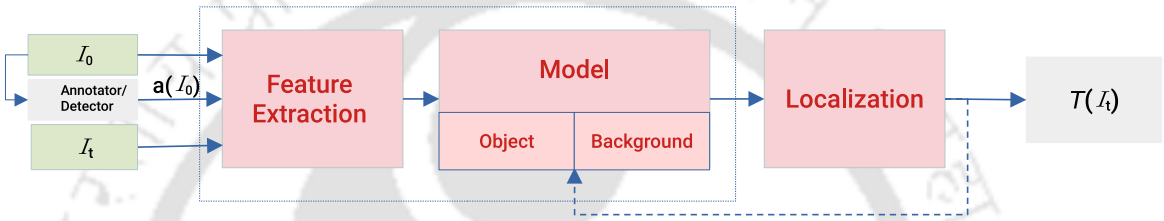

Figure 2.3: Block diagram of a classical tracking algorithm.

The classical approaches use hand-crafted features, learns statistical and/or machine learning models on the extracted features and use localization strategies [1, 27]. This can be illustrated by a block based modular approach as shown in Figure 2.3. In the literature, tracker proposals from this group are found to make contributions in one or more of these blocks. Two important blocks are the feature extraction and object modeling. The simplest feature can be image pixel intensity [4]. The derived features can be more complex and include Scale Invariant Feature Transform (SIFT) [28], Oriented FAST and Rotated BRIEF (ORB) [29], Fast Retina Keypoint (FREAK) [30], sparse codes of patches [31] among others. They have certain desired qualities like robustness to certain challenging scenarios that could be present in a tracking sequence like geometric transformations, illumination change and others. Statistical modeling as well as machine learning based models exist in the literature. A good object model contributes to a successful tracker [32]. Sometimes background is also modeled along with object to improve discrimination. Based on object modeling, algorithms are classified into generative, discriminative and hybrid approaches.

Generative approaches construct object models from features like color distribution [4], contour [21], object template [33], pixel motion distribution [34], Histogram of Oriented Gradients [31] among others. Machine learning based models are mostly seen in discriminative approaches where the objective is to learn characteristics of object that differentiate it from the background. Such ap-

proaches are realized with Support Vector Machines [35], Multiple Instance Learning [36], P-N learning [37] among others. Tracking is usually performed as a search to find a region that maximizes similarity with the object model. Here, the object features are mostly exploited. In contrast, a discriminative approach identifies characteristics that differentiate an object from its background. Detectors or classifiers are used for this purpose. Object localization is performed by searching for a region that maximizes a similarity metric or classification score computed from the target and candidate models. A hybrid approach have both generative and discriminative components. Tracking algorithms based on different object models are shown in Figure 2.7. Different classical approaches that use the generative models are discussed next.

2.2.1 Generative Models

The generative models are constructed from the object characteristics. Features like pixel intensity values [33], edges [34], motion [38], derived features [28], sparse codes [9] are used to build these models.

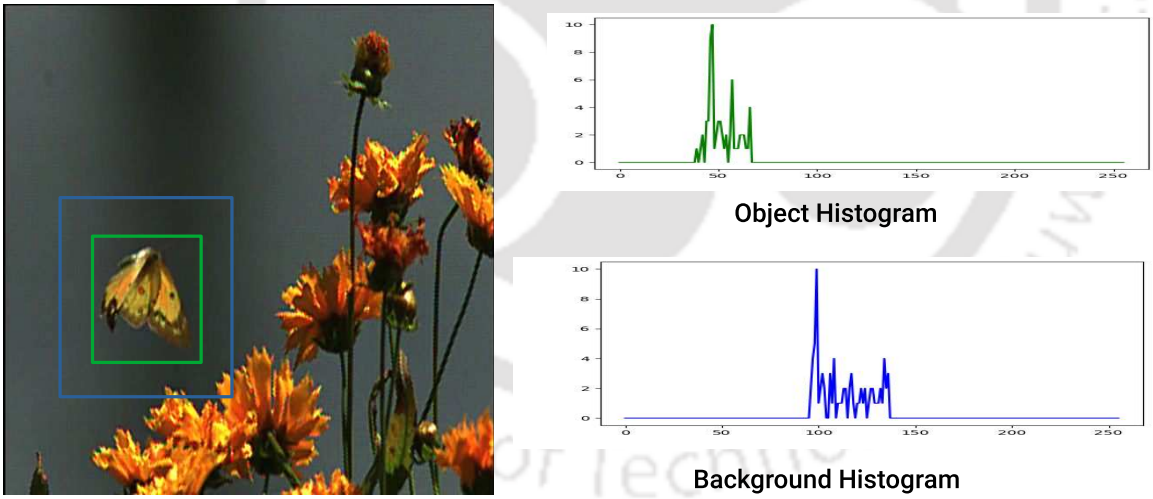

Figure 2.4: Color Histogram of object and its immediate background

One of the most popular and simple object model is the color distribution. A kernel weighted color distribution used by Comaniciu *et al.* [4] in their proposal on mean-shift tracking [4]. Here, the target object is modeled from the color distribution in its bounding rectangle $\mathbf{bb}_0^+$ as follows

$$\mathbf{m}_0^{\mathbb{T}+}[u] = L \sum_{i=1}^{n_x} k(\|\overset{\star}{\mathbf{x}}_i\|^2)\delta[b(\overset{\star}{\mathbf{x}}_i) - u] \tag{2.5}$$

where, $b(\overset{\star}{\mathbf{x}}_i)$ computes the bin index corresponding to the pixel intensity value at location $\overset{\star}{\mathbf{x}}_i$, n_x is the

total number of pixels in bounding box $\mathbf{bb}_0^+$, $\delta[.]$ is the Kronecker delta function and $k(\mathbf{x})$ is the kernel profile function that weighs the pixels with their relative distance from center. The normalization constant L ensures that $\sum_{u=1}^{n_b} \mathbf{m}_0^{\mathbb{T}+}[u] = 1$ with n_b number of bins in the quantized feature space.

Similarly, the candidate region has the kernel with profile function $k(\mathbf{x})$ and bandwidth h whose support is centered at $\mathbf{y}$. The candidate model $\mathbf{m}_t^{\mathbb{C}+}(\mathbf{y})$ is thus defined as the feature distribution (normalized with the constant L_h) computed from the pixel locations $\{\mathbf{x}_i\}$ $(i = 1, \ldots n_x)$ and is given by

$$\mathbf{m}_t^{\mathbb{C}+}(\mathbf{y})[u] = L_h \sum_{i=1}^{n_x} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right) \delta[b(\mathbf{x}_i) - u] \tag{2.6}$$

where, n_x is the total number of pixels in $\overset{\star}{\mathbf{bb}}_{t-1}^{+}$.

Above model is robust against scaling, rotation and can handle object deformation. However, histogram being a global feature, loses information on object structure. To overcome this, Peter *et al.* [39] divided an object into parts and histograms were learned from each part. Such a multi-part approach could capture the spatial layout of the object. The performance of a color feature based tracker may get affected in two ways. Same pixel intensity values of object might be present in a scene background. Also, some background pixels might be present in an object bounding rectangle (for non-rectangular object). This can corrupt an object model. To improve robustness, pixels that belong only to the object are to be given more importance during modeling. A form of this can be seen with Comaniciu *et al.* [4] where object pixels closer to object centroid are weighted more compared to the ones at periphery. This way of weighing is purely based on distance and not on probability of a pixel belonging to the object. Jifeng *et al.* [40] proposed a background weighted histogram model where weights of each color bin of an object model is dependant on its contribution in background model. Comaniciu *et al.*estimated scale by repeating the procedure at predefined values. Efficient estimation of object scale was proposed by Robert *et al.*in [41]. Zivkovic *et al.* [42] proposed scale and orientation estimation from covariance matrix by using a Gaussian kernel. Objects with similar color distribution can be present in the background that can interfere in localization. Thus, explicit modeling of distractor was performed by Horst *et al.*in [43]. Changjiang *et al.* [44] proposed a novel similarity measure for histograms. Color Histogram model generation is shown in Figure 2.4. To improve robustness of object model, the derived features are explored in the literature. One such popular approach is the sparse representation of object template for tracking.

Mei *et al.* [23] proposed the first notable work on sparse representation based target model. Target

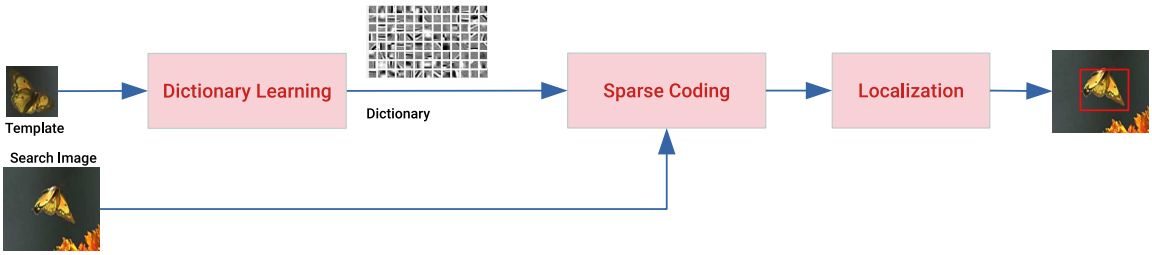

Figure 2.5: Block diagram of a sparse representation based tracker. Dictionary is learned from the object template. Sparse codes are computed for candidate regions from the search image $\mathcal{I}_t^S$ during localization.

tracking was formulated as a ℓ_1 minimization problem where the candidate model was represented as a sparse linear combination of object and trivial templates. A basis set consisting of object templates and trivial templates were respectively used to model appearance and partial occlusion. Candidate template regions were sampled in a particle filter framework. The best candidate solution was identified as the one that minimizes reconstruction error with the target templates. The tracking problem was formulated as

$$\min \|\mathbf{D}\boldsymbol{\gamma} - \mathbf{p}_{ti}\|_2^2 + \lambda \|\boldsymbol{\gamma}_{ti}\|_1 \tag{2.7}$$

where, λ is the regularization coefficient, $\boldsymbol{\gamma}_{ti}$ is the i^{th} candidate region image vector, $i = 1, \ldots, n_{\mathbb{B}}$, $\mathbf{D}$ consists of target and trivial templates, $\mathbf{p}_{ti}$ is a potential candidate template vector and $\boldsymbol{\gamma}_{ti}$ is its sparse code representation with respect to the dictionary $\mathbf{D}$. This is solved as ℓ_1 regularized least squares problem [23]. A generic block diagram of a sparse representation based tracker is shown in Figure 2.5

Tianzhu *et al.* [45] proposed joint learning of candidates to make use of the structural relationship among candidates. Online dictionary update was proposed by Baiyang *et al.*in [46]. Also, it highlighted the importance of discrimination power of a dictionary in tracking as compared to its reconstruction power. Only few works have used background information in learning representations. Discriminative information computed from foreground and background atoms were used in [47]. Bai *et al.* [48] used background information to eliminate outlier candidates based on their sparse codes. Sparse representation based approaches use learning from data. However, it is mostly limited to the dictionary learning. Here, statistical models of sparse codes were mostly used to represent the object.

2.2.2 Discriminative Models

Discriminative methods approach tracking as a binary classification problem where object and its background form the two classes. This approach identifies and learns features that differentiates an object from its surroundings. These features range from pixel intensity values to more complex derived ones. The most popular discriminative approach is based on correlation filter learning [49]. Discriminative filter learning is explored in correlation based trackers. Many machine learning classifier based algorithms are also used for object modeling in literature.

Kaihua *et al.*proposed a Bayes classifier for tracking in [50] with features extracted from multiple scales. Kalal *et al.* [37] proposed the use of P-N learning in a tracker. Multiple instance leaning based classifier was used in [51] etc.. Grabner *et al.* [52] proposed online training of classifiers to adapt to object appearance changes. Kalal *et al.* [53] proposed computation of forward and backward error to identify consistent trajectories. Large motions are handled with image pyramids in a coarse to fine manner. Classifiers were trained online to adapt the tracker to object appearance changes in [52]. Ross *et al.* [54] learned object changes incrementally with principal component analysis. Grabner *et al.* [55] proposed a self-supervised online boosting to address the problem of tracker drift. Online update was proposed to counter tracker drift in [56]. Martin *et al.* [57] proposed the use of online Hough forests for tracking as well as segmentation of the object. Stefan *et al.* [58] proposed co-training of detector and segmentation approach to achieve non-rigid tracking.

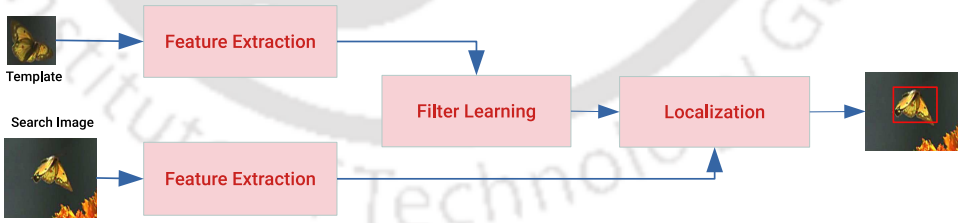

Figure 2.6: Block Diagram of a correlation filter based tracker. Correlation filter coefficients are learned from the object template image $\mathcal{I}^Q$

Significant work has been done in tracking with discriminative correlation filters (DCF) [49]. The oldest and simplest example of such an approach is seen with template tracking. Correlation tracker learns filters on the initialized object template where all possible shifts of the object template is used for learning. It uses computations in Fourier domain to achieve high speed. Let $\{(\mathbf{X}_j, y_j); j = 1, \ldots, n_x\}$ be the n_x training samples. $\mathbf{X}_j \in \mathbb{R}^{M \times N}$ is the j^{th} single channel feature map from an image region. A spatial location is represented by $(m_1, m_2) \in \Omega := \{1, \ldots, M\} \times \{1, \ldots, N\}$. $\mathbf{X}_j(m_1, m_2) \in \mathbb{R}$ rep-

resents the scalar value at location (m_1, m_2). y_j is a scalar valued function defined over the spatial locations $(m_1, m_2) \in \Omega$. Usually a Gaussian function with a peak at the centroid location of object is used. Basic single channel formulation of a discriminative correlation filter $\mathbf{W}^{cf} \in \mathbb{R}^{M \times N}$ is learned such that

$$\mathbf{X}_j * \mathbf{W}_j^{cf} \approx y_j \tag{2.8}$$

The DCF formulation uses convolution operations and is given as follows.

$$\mathbf{X} * \mathbf{W}^{cf}(m_1, m_2) = \sum_{k=1}^{M} \sum_{l=1}^{N} \mathbf{X}((m_1 - k), (m_2 - l)) \mathbf{W}^{cf}(k, l) \tag{2.9}$$

In the pioneering work by Bolme *et al.* [59], a Minimum Output Sum of Square Error (MOSSE) filter was proposed for a high speed correlation tracker with robustness to scale, lightning and pose. Further improvement was brought in by removing the redundancies of overlapping patches by exploiting the circulant nature of the data matrix in [60]. Yang *et al.* proposed a scale adaptive learning paradigm in [61]. Martin *et al.* [62] learned independent correlation filters to handle translation and scale. A re-detection strategy along with object template and context was proposed to achieve long term tracking in [63]. Martin *et al.* [64] proposed a spatial regularization component to mitigate the boundary problems associated with the correlation filters. Similarly, a spatio-temporal regularization to the filters were introduced by Feng *et al.* [65]. Matthias *et al.* [66] proposed a context aware approach to minimize the effect of distractors in tracking. Bibi *et al.* [67] proposed joint learning of filter and target response to counter tracker drift. An ensemble with key-point matching based tracker was proposed in [68]. Bertinetto *et al.* [69] proposed a combination of color and correlation filter for robustness against object deformation. Alan *et al.* [70] proposed spatial and channel reliability maps in learning filters to achieve long term tracking. More on correlation based trackers can be found in [49]. Recently, correlation filters are learned on the feature embeddings generated by deep networks. More on these are reviewed in section 2.3.

2.2.3 Model Update

While the majority of current algorithms can successfully track object in controlled surroundings, they frequently fall short when there is a considerable variation in the object's appearance or ambient lighting. Numerous algorithms use fixed appearance representations of the target, which could be one

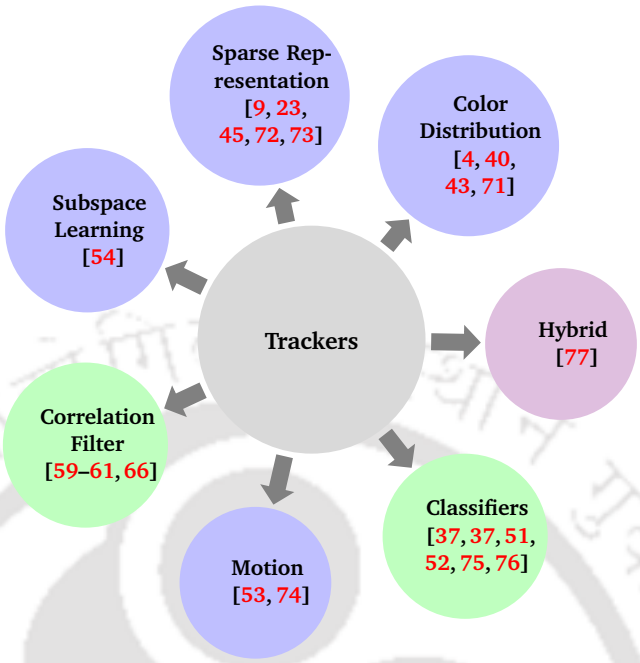

Figure 2.7: Statistical and Machine Learning based trackers with generative, discriminative and hybrid object modelling. Generative approaches are shown in blue color, disriminative ones in green and hybrid ones in violet

cause of these failures. An object model requires update as tracking progress to capture the changes it undergoes. It is a tricky thing to do model update as too frequent update can result in corruption of the model and can result in tracker drift. On the other hand, very few update means it will fail to capture the transformations undergone by the object. David *et al.* [54] proposed incremental learning of low dimensional object subspace with Principal Component Analysis (PCA) to adapt to the appearance changes of object. Matthews *et al.* [78] proposed a template update procedure without causing tracker drift. Grabner *et al.* [52] used online training of classifiers to adapt the tracker to object appearance changes. Model update procedure based on object covariance matrix was proposed in [79].

2.2.4 Localization

During this phase the target is located in the given frame $\mathcal{I}_t, t \geq 1$. This stage usually involves a search operation to locate the object. Different search techniques like motion models [80], iterative search [27, 41], probabilistic sampling [21, 23], meta-heuristic approaches [24, 25, 81] and others are explored in the literature. A naive approach would be to perform a brute force search in the entire frame or in a region of interest. Another approach is to use candidate region proposal(s) and

iteratively identify the best among them. Some of these approaches are shown in Figure 2.8. The search algorithms optimize an objective function for locating the target in $\mathcal{I}_t$. It is common in both generative and discriminative approaches to define the objective function based on the models. Object localization searches for a region that maximizes (or minimizes) a similarity (or dissimilarity) measure or maximizes a classification score.

Figure 2.8: Illustration of different localization approaches. Figure shows (a) brute-force, (b) iterative and (c)sampling based approaches. Brute-force approach goes through all the locations in the image. Iterative search starts with an initial region $\overset{\star}{\mathbf{bb}}{}^{+}_{t-1}$ (red box) and performs a directed search to converge at an estimated $\overset{\star}{\mathbf{bb}}{}^{+}_{t}$ (green box). Sampling based approach has multiple box initalizations $\mathbf{bb}_{ti}$ which are updated iteratively till convergence.

Yilmaz *et al.* [2] used a brute force approach to perform template matching. Let $\mathbf{p}^{Q}$ be the template patch vector and $\mathbf{p}_{ti}$ be the candidate patch vector defined from the region proposal $\mathbf{bb}_{ti}$ and search image $\mathcal{I}_t^S$. A brute force approach for estimating $\overset{\star}{\mathbf{bb}}{}^{+}_{t}$ is defined as follows

$$\overset{\star}{\mathbf{bb}}{}^{+}_{t} = \mathbf{bb}_{ti^\star}; i^\star = \underset{i=1,\ldots,n_{\mathbb{B}}}{\arg\min} \|\mathbf{p}^{Q} - \mathbf{p}_{ti}\| \tag{2.10}$$

where, $n_{\mathbb{B}}$ is the total number of region proposals. Comaniciu *et al.* [82] proposed the mean-shift iterations for target localization (Equation 2.11). The search is initialized in the current frame $\mathcal{I}_t$ using the object position from the previous frame $\overset{\star}{\mathbf{bb}}{}^{+}_{t-1}$. The object localization starts from $\overset{\star}{\mathbf{bb}}{}^{+}_{t-1}$ $\hat{\mathbf{y}}_0$ in the current frame and iteratively tends to maximize the Bhattacharyya coefficient based similarity $\rho^{+}(\mathbf{m}_0^{\mathbb{T}+}, \mathbf{m}_t^{\mathbb{C}+}(\mathbf{y}))$ between the target and candidate models. The mean-shift centroid update rule from $\mathbf{y}_\tau$ in τ^{th} iteration to $\mathbf{y}_{\tau+1}$ is given by

$$y_{\tau+1} = \frac{\sum_{i=1}^{n_x} w_i^{(\tau)} \mathbf{x}_i g\left(\left\|\frac{\mathbf{y}_\tau - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_x} w_i^{(\tau)} g\left(\left\|\frac{\mathbf{y}_\tau - \mathbf{x}_i}{h}\right\|^2\right)} \tag{2.11}$$

where $g(x) = -k'(x)$ is the negative derivative of the kernel profile function $k(x)$, $\mathbf{x}_i$ is the pixel position within the bounding box and the weights $w_i^{(\tau)}$ are given by

$$w_i^{(\tau)} = \sum_{u=1}^{n_b} \sqrt{\frac{\mathbf{m}_0^{\mathbb{T}+}[u]}{\mathbf{m}_t^{\mathbb{C}+}(\mathbf{y}_\tau)[u]}}\, \delta[b(\mathbf{x}_i) - u] \tag{2.12}$$

Here, $\mathbf{m}_0^{\mathbb{T}+}$ is the target object model and $\mathbf{m}_t^{\mathbb{C}+}$ is the candidate model and $\delta(\bullet)$ is the Kronecker delta function.

Sampling based approaches use region proposals to generate potential candidates. Such methods are seen with condensation algorithm [83], evolutionary, and swarm based search strategies. These algorithms are useful when it is not possible to compute the gradient. Also, meta-heuristic algorithms were found to provide faster convergence in certain cases [24]. These algorithms generate solutions by following a set of rules in a random yet directed manner. This directed approach leads to better convergence and the randomness helps in search space exploration (or exploitation). Some of these approaches include Genetic Algorithm (GA) [84], Particle Swarm Optimization (PSO) [6, 85], Firefly Algorithm (FA) [86] etc.. Patrick *et al.* [39] proposed the use of Monte Carlo sampling with a color histogram model. Tracking was formulated as a ℓ_1 minimization problem with particle filter approach in [23]. Zhang *et al.* [87] formulated tracking as an entropy minimization problem. Bounding box proposals were generated based on edge information over the entire frame rather than in a search window in the proposal [88].

Inter-frame motion is an important cue in localization. Motion captures information of the dynamic behaviour of the object or the camera. Motion analysis is the basis of many high level tasks, including tracking. Optical flow techniques are used for motion estimation in classical approaches [89, 90]. Optical flow refers to motion of the intensity pattern from one frame to other. This is the 2D motion on the image frame and is different from the 3D motion in a scene. Motion is estimated between two frame $\boldsymbol{\mathcal{I}}_t$ and $\boldsymbol{\mathcal{I}}_{t+\tau}$ that are apart in time by τ. The motion field $\mathcal{O}^F \in \mathbb{R}^2$ provides components of motion along horizontal (x) and vertical (y) directions. It is computed with the brightness constancy constraint equation which assumes that the pixel intensity remains same during displacement. It is

given as follows.

$$\mathcal{I}(\mathbf{x} + \mathcal{O}^F(\mathbf{x}), t+1) = \mathcal{I}(\mathbf{x}, t) \tag{2.13}$$

Here, $\mathbf{x}$ is the pixel location and t is the time. The displacement is considered small for optical flow estimation [91]. Large displacements are handled with image pyramids in a coarse to fine approach [74, 92, 93]. Classical approaches to dense optical flow computation can be found in proposals of Gunnar-Farneback [94] and Horn–Schunck [89]. Recently, deep neural network based methods were developed to compute inter-frame dense motion (See section 2.3). Predicting the motion of all object pixels from one frame to the next frame can localize the object. Santner *et al.* [77] used mean-shift algorithm with optical flow information to perform the tracking. Kalal *et al.* [53] proposed the median flow approach in a forward-backward framework to estimate the target position. A modification was proposed by Han *et al.* [95] with RANSAC algorithm to handle occlusions. Okada *et al.* [96] proposed the use of motion histograms to locate the object. Sanjivani *et al.* [97] used optical flow for object localization and Kalman filter for tracking. Connected regions with similar flow vectors were identified to perform tracking in [98].

2.2.5 Ensemble Approaches

Ensemble techniques are widely used in machine learning to improve over the performance of component base classifiers. The same is realized in different tracking approaches. Combinations are proposed in the feature extraction, model construction, and localization stages. Also, independent trackers are combined to form an ensemble. In an ensemble [99], Weak learners are combined to create a strong learner [55]. Ye *et al.* [100] showed that the diversity is an important factor for a successful ensemble. A combination of complementary components can improve overall performance of an ensemble [77, 101].

Ensembling can be performed at the feature extraction and modeling stage or during the localization stage. A popular ensemble approach with the object model is the use of object parts rather than a single template. The object is usually divided into parts and are tracked independently. For example, Adam *et al.* [102] proposed the use of histogram to model individual parts of the object. The final object location is estimated by combining (e.g. weighted average) the predictions of independent object-part trackers.

Kwon *et al.* [103] proposed the use of multiple appearance models in sparse representation framework. An ensemble of weak classifiers are combined using AdaBoost in [75]. Tian *et al.* [104] proposed an ensemble of SVM classifiers to handle appearance changes with the use of historical information.

A hybrid object model uses both generative and discriminative components [77]. Bertinetto *et al.* [69] proposed a combination of color and correlation filter to provide robustness against object deformation. S. Avidan *et al.* [35] integrated SVM based classifier with optical flow based tracker.

A single tracker might not be able to handle all challenging scenes. Each of them might be good at handling certain aspects of tracking. Several proposals have been made in the literature to combine individual trackers in an ensemble [105, 106]. Kwon *et al.* [105] proposed an ensembling of trackers which selected an optimum tracker from a predefined set of trackers. Senna *et al.* [106] proposed the fusion of the output of various trackers with Kalman filter. This approach tries to combine the strengths of each of the individual trackers in the ensemble. Zikun *et al.* [107] proposed switching between local trackers (upon failure) to achieve long-term tracking. Gao *et al.* [108] fused the outputs of multiple trackers executed in parallel to achieve optimal performance.

TIMELINE 1: *Classical Approaches in Tracking*

- 1996 Conditional Density Propagation and Applications to Visual Tracking [21]
- 1998 Real time face and object tracking as a component of a perceptual user interface [109]
- 2000 Real-time tracking of non-rigid objects using mean shift [110]
- 2000 Learning patterns of activity using real-time tracking [111]
- 2003 Mean-shift blob tracking through scale space [41]
- 2004 Support vector tracking [35]
- 2004 The template update problem [112]
- 2005 Online selection of discriminative tracking features [113]
- 2006 Object tracking: A survey [2]
- 2006 Real-time tracking via on-line boosting [52]
- 2007 Ensemble tracking [75]
- 2008 Incremental learning for robust visual tracking [54]
- 2008 Semi-supervised on-line boosting for robust tracking [55]
- 2009 Visual tracking with online multiple instance learning [51]
- 2010 Forward-backward error: Automatic detection of tracking failures [53]
- 2010 Visual object tracking using adaptive correlation filters [59]
- 2010 Robust object tracking with online multiple instance learning [36]
- 2011 Tracking-learning-detection [37]
- 2012 Real time robust L1 tracker using accelerated proximal gradient approach [9]
- 2012 Real-time compressive tracking [50]
- 2012 Robust object tracking via sparsity-based collaborative model [114]
- 2012 Exploiting the circulant structure of tracking-by-detection with kernels [115]
- 2012 Robust object tracking via sparsity-based collaborative model [114]
- 2012 Visual tracking via adaptive structural local sparse appearance model [116]
- 2013 Online object tracking: A benchmark [117]
- 2013 Learning a deep compact image representation for visual tracking
- 2014 Accurate scale estimation for robust visual tracking [118]
- 2014 A scale adaptive kernel correlation filter tracker with feature integration [61]
- 2014 High-speed tracking with kernelized correlation filters [60]
- 2014 MEEM: robust tracking via multiple experts using entropy minimization [87]
- 2014 Adaptive color attributes for real-time visual tracking [119]
- 2015 Hierarchical convolutional features for visual tracking [120]
- 2015 Struck: Structured output tracking with kernels [76]
- 2015 Long-term correlation tracking [63]
- 2016 Learning spatially regularized correlation filters for visual tracking [64]
- 2016 Staple: Complementary learners for real-time tracking [69]

2.3 Deep Neural Network based Approaches

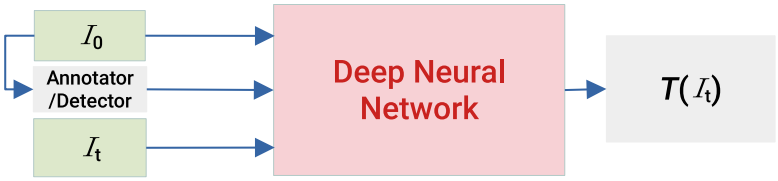

Figure 2.9: Trackers formulated in Deep Learning framework.

Fukushima [121] introduced Artificial Neural Networks (ANN) with a neocognitron model in 1980. LeCun *et al.* [122] used a back propagation based technique to train a multi-layer fully connected network with three hidden layers for hand written ZIP code recognition. However, they were restricted by the large training time. Hinton [123] proposed a fast multi-layer deep belief network (DBN) trained with back propagation. Such networks existed in the literature but, were not the dominant methods in computer vision until recently. It was AlexNet's [124] success that re-ignited the interest in deep neural networks (DNNs). Availability of powerful computational hardware and large corpus of data aided in developing better architectures. The popular ones include Convolutional Neural Networks (CNNs). Recurrent Neural Networks (RNNs), Long Short Term Memory Networks (LSTMs) etc.. The DNNs have found success in many vision related tasks and outperformed the classical approaches [125]. Object tracking is no exception and current literature is dominated by DNN based tracking algorithms [24].

The classical approaches rely on hand-crafted features, statistical models or machine learning techniques. In most cases, a target model is constructed in the initialization stage. During tracking, the target localization is performed by comparing target and candidate models. In contrast, the deep neural network based trackers promote an end-to-end approach to tracking (See Figure 2.9) with learned features. However, in the early adoptions, the DNNs were used as feature extractors only and then coupled with other statistical modeling and localization techniques [120,126]. These approaches were limited by their execution time for tracking. The end-to-end trackers are much more efficient and are able to provide better performance [127]. The DNN based tracking algorithms are broadly divided into the following five categories – (a) matching based, (b) learning based, (c) sequential, (d) attention based and (e) ensemble approaches.

Rest of the section is organized as follows. Sub-section 2.3.1 describes the matching based trackers while mostly discussing the siamese trackers. Sub-section 2.3.2 describes the deep correlation filter based approaches. Subsection 2.3.3 discusses trackers that exploit temporal information. Sub-

section 2.3.5 describes some of the learning based approaches. Subsection 2.3.4 briefly discusses the recent transformer based approaches.

2.3.1 Matching based: Siamese Network Trackers (SNTs)

Siamese network (SN) has a Y-shaped architecture (Figure 2.11) with two branches that are joined to produce a single output [128]. It is trained to predict the similarity of its two inputs. Generally the two branches have shared weights among them. Higher output value indicates that the inputs are similar. Such a network is trained on image triplets. A triplet has an anchor image ($\mathcal{I}^A$), positive image ($\mathcal{I}^P$) and a negative image ($\mathcal{I}^N$). Positive image is one which is similar to the anchor and the negative image is the one that is dissimilar to the anchor. On training the network, it makes the embeddings of positive pairs closer and the embeddings of the negative pair apart at least by a margin hyper-parameter η_m. Effect of similarity learning on the feature distance is shown in Figure 2.12. It is then trained with a triplet loss defined as follows [129].

$$\mathcal{L}(\mathcal{I}^A, \mathcal{I}^P, \mathcal{I}^N) = \max(\|f(\mathcal{I}^A) - f(\mathcal{I}^P)\|^2 - \|f(\mathcal{I}^A) - f(\mathcal{I}^N)\|^2 + \eta_m, 0) \tag{2.14}$$

The SNs were first applied in the signature verification task [130]. Object tracking is formulated as a matching problem with similarity learning in a siamese network [131]. However the matching should be able to handle the variations that an object can undergo in a sequence. The SNTs belong to a family of matching based trackers where feature embeddings are compared [101]. From the study performed by [132–134] it was shown that due to their ability to balance between performance and efficiency, siamese-based networks are the most promising designs in tracking. Many of the recent Siamese trackers are also fully convolutional (FC) [10] in nature allowing them to work with inputs of variable dimensions. Among other things, this allows for an end-to-end training network to be designed with fewer hyper-parameters [135].

The SNTs learn a similarity function that provides the similarity between the embeddings of its inputs. These embeddings are computed by a shared sub-network. One of the input is the template of the target object. The other input is the search image on which the target is to be localized. The SNTs work in two stages. The first stage involves the feature extraction (embedding) of the inputs using CNN sub-networks. These embeddings are compared in the second stage. Bertinetto *et al.* [10] proposed a fully convolutional siamese (SiamFC) tracker with cross-correlation operation performed

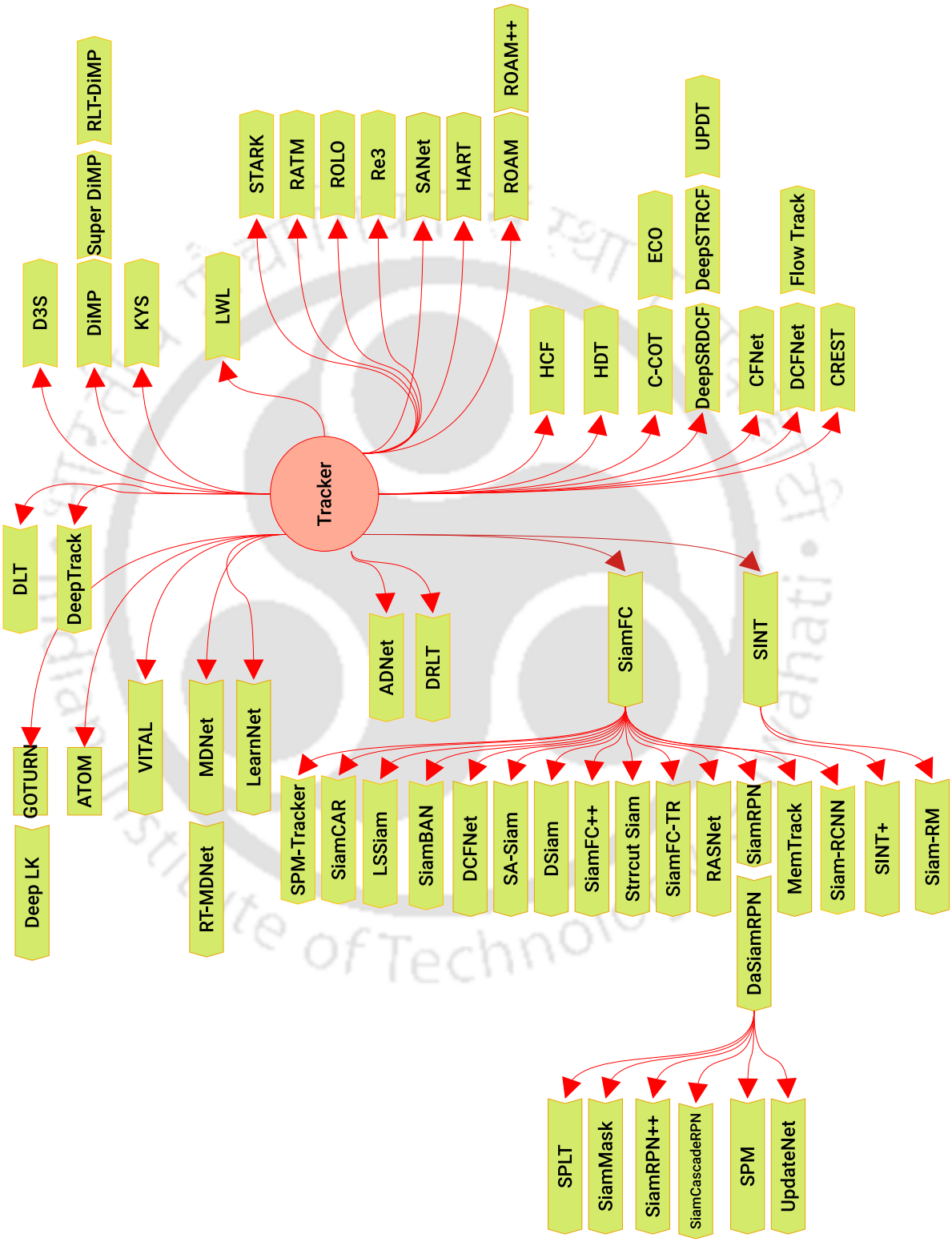

Figure 2.10: Deep Learning based Trackers

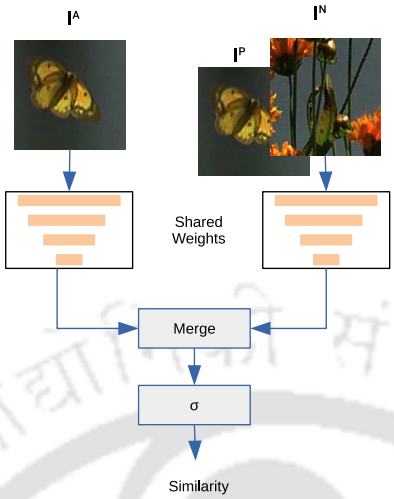

Figure 2.11: A Siamese network for similarity learning. The inputs to the network are anchor image $\mathcal{I}^A$ and a similar looking image $\mathcal{I}^P$ and a dissimilar looking image $\mathcal{I}^N$ in pairs

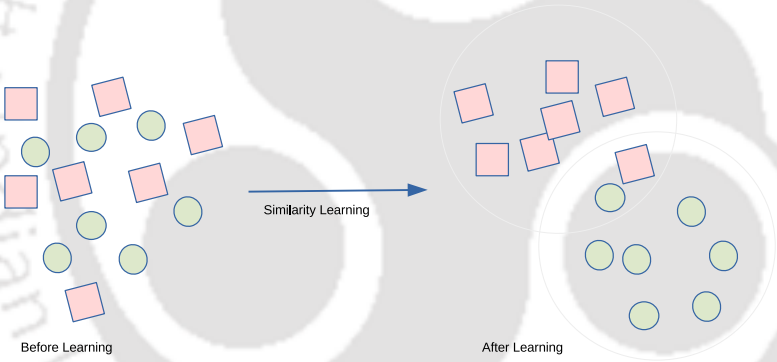

Figure 2.12: A Transformation with Similarity Learning. Similar inputs are brought together in the embedding space after training while different ones are pushed away.

on the embeddings. Bertinetto *et al.* [10] showed the commutable property of a fully convolutional network. Let G represent a translation such that $G_\tau x[t] = x[t-\tau]$, then function f is fully convolutional with integer stride m if

$$f(G_{m\tau}x) = G_\tau f(x) \tag{2.15}$$

This allows for similarity search in sub-windows in one pass. This property is termed as translation equivariance [136] and is one of the reasons for the success of similarity learning in object tracking. This fully convolutional property is widely used in many of the subsequent trackers.

Tao *et al.* [137] proposed the first siamese tracker which formulated it as a similarity search prob-

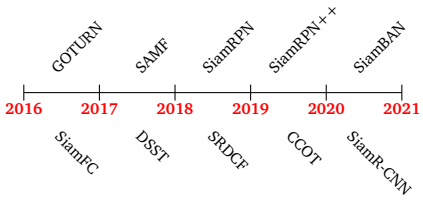

Figure 2.13: A timeline of Siamese Network trackers (SNTs)

.

lem. Initial SNTs used a pre-trained AlexNet [124] after fine-tuning [10, 137, 138] the backbone network. Here, the performance was limited as AlexNet is a relatively shallow network and was not able to generate sufficiently rich features [139]. Later SNT proposals switched to deeper and wider networks that are based on ResNet [140], VGG-19 [14], Inception [141] networks among others. Deeper layers produced richer semantic features. Li *et al.* [139] improved performance with a custom VGG16 [14] backbone. However, Li *et al.* [142] showed that direct replacement with deeper networks is not sufficient to improve performance. Data bias in pre-trained networks trained on ImageNet affect tracking performance [143].

The SiamFC tracker is not efficient in estimating the accurate scale of the object as it performs a multi-scale search over the image. Anchor based and anchor-free approaches are explored to handle scaling of object. Jia *et al.* [144] made the tracker scale adaptive with a region proposal network (RPN) [145]. This also resulted in an increased number of hyper-parameters. A version with ResNet backbone is proposed in [142]. Guo *et al.* [146] avoided anchor boxes of RPN and used bounding box regression and classification at pixel level to have a tracker with lesser hyper-parameters. Chen *et al.* [147] proposed a similar approach with atrous convolutions. Instead of final layer cross-correlation, Guo *et al.* [148] proposed a multi-level feature extraction and cross-correlation. Wang *et al.*. [149] proposed a binary segmentation mask of the object while using depth-wise cross-correlation on the multiple score maps.

Structural information captured with embeddings might not help in the presence of similar looking objects a.k.a distractors in background. Zhu *et al.* [150] proposed hard negative mining in training to improve robustness to distractors. Li *et al.* [151] addressed the same with a re-detection module which is an online classifier in general. Voigtlaender *et al.* [152] extended it further with a two stage detector model. Multi-level feature is commonly seen with many SNTs [13, 146]. Lian *et al.* [153] highlighted that learning local features aids in tackling drift. To handle occlusion, Wu *et al.* [154] proposed

a tracker with anti-occlusion region proposal network. Liu *et al.* [155] proposed a multi-template approach to handle appearance changes of object. Pflugfelder [128] highlighted in their studies that SNTs require additional mechanisms like attention, regression etc. to improve performance.

2.3.2 Filter Learning: Discriminative Filter approach

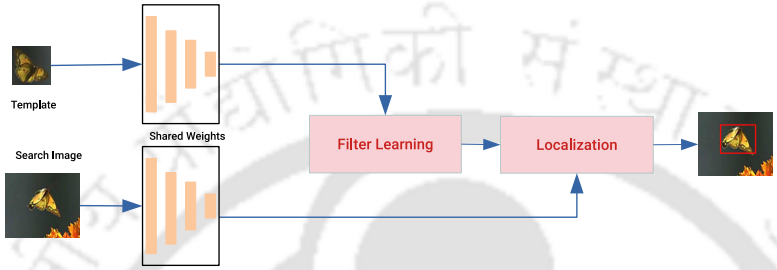

Figure 2.14: Deep correlation filter based tracking. Filter is learned on the feature embeddings generated by network $\Phi(\bullet)$ rather than on intensity values as seen with classical approaches.

Classical approaches with discriminative correlation (DCF) filter relied on intensity and hand-crafted features for learning. Extension of the same which works on feature embedding [14, 140] generated by a DNN is proposed in literature (Figure 2.14). Chao *et al.* [120] learned correlation filter on each of the hierarchical convolution layers to track the object. Ma *et al.* [120] proposed generating correlation maps from multiple layers of the DNN. Shallow layers were given relatively more importance compared to the deeper ones. Similar strategy is followed by [126]. Deeper layer features have more semantic information but are less invariant to object changes like deformations, rotation etc.. On the other hand, shallow layer features are much more invariant to transformations [64]. Therefore it becomes important to combine the information from different layers to improve tracking [156, 157]. As seen with the classical approaches, DCF use Fourier transform to compute the predictions and are affected by boundary effects. There are trackers that are designed to mitigate this artefact with regularization in the filter learning process. Spatial regularisation was proposed with approaches like [158]. Li *et al.* [65] proposed a temporal regularization in learning. Xu *et al.* [159] proposed channel regularization along with spatial one. Coarse resolution features with custom learners were employed in [160, 161] to address the boundary effects. More on correlation filter based trackers can be found in [162]. A time line of advancement of correlation based trackers is shown in Figure 2.15.

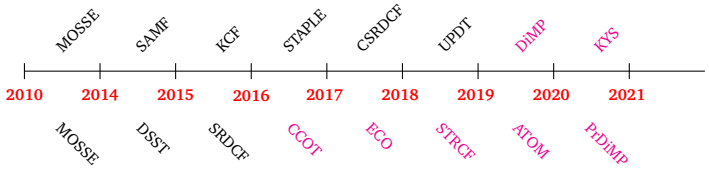

Figure 2.15: A timeline of discriminative correlation filter based trackers. Trackers following the classical approach are in blue color and trackers based on DNNs are shown in magenta color.

2.3.3 Temporal Approaches: RNNs & LSTMs

Temporal aspect of a sequence is often discarded in the CNN based approaches [134]. The Recurrent Neural Networks (RNNs) are effective in learning sequential information but are restricted by the vanishing gradient problem. To overcome this, the LSTMs were introduced. The RNNs can model not only temporal but also spatial information [163]. The contextual relationship of Objects in the spatial domain is an important yet underutilized property in appearance modeling [163]. Quan *et al.* [163] proposed an end-to-end trainable recurrent neural network that transformed the object context into semantic sequential representation. A recurrent neural network was used for this within a LSTM network. The target object was represented as a series of overlapping image patches. Each patch in the sequence was assigned a label by the RNN during tracking.

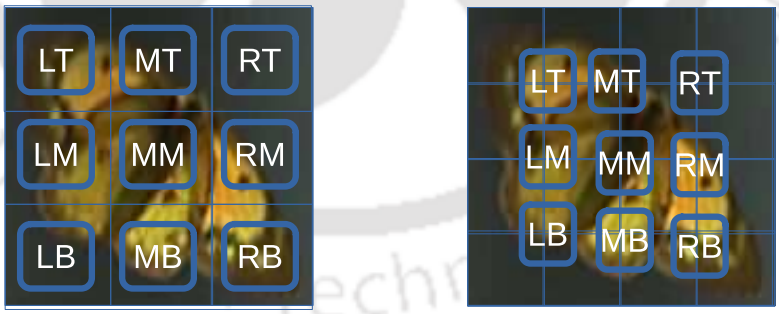

Figure 2.16: Target as a sequence of patches as given in [163]. On the left is the representation without overlap and on right is the representation that has overlap. The patches are labeled as left-top (LT), left-middle (LM), left-bottom (LB), middle-top (MT), middle-middle (MM), middle-bottom (MB), right-top (RT), right-middle (RM), right-bottom (RB).

Fan *et al.* [164] brought distractor awareness with RNNs to a CNN based approach. Guanghan *et al.* [165] looked at features from tracking history to build a robust tracker with LSTM network. The LSTM was used along with CNNs to exploit spatio-temporal information. Historical position and visual features were used in addressing tracker failures.

Cui *et al.* [166] addressed tracker drift by identifying reliable parts of the object with a multi-directional RNN. Kosiorek *et al.* [167] proposed an attentive model that can differentiate the object from its background. A similar soft attention based approach was proposed by Samira *et al.* [168]. Real time tracking with online update of object appearance was proposed by Gordon *et al.* [169]. A combination of LSTM and YOLO detector [170] was proposed by Ning *et al.* [165] to estimate target location. Learning directly on video segments for tracking was proposed in [171].

2.3.4 Attention based Approach

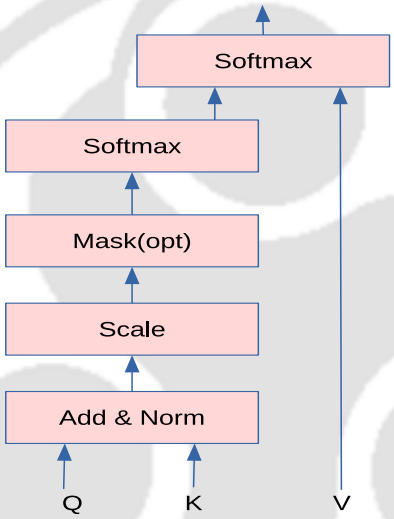

Figure 2.17: Scaled dot product based attention in transformer. Q,K,V are the query, key and value respectively as seen with image retrieval systems

Transformers [172] were introduced in natural language modeling and were then explored in representation learning for vision tasks [173–175]. Recently transformer [172] based approaches were successfully proposed in visual object tracking. It is used as a feature extractor [175] or as an end-to-end network in tracking [134]. They can be combined with existing trackers like siamese and correlation filter based ones. Ning *et al.* [176] proposed a transformer where encoder and decoder networks were used in a siamese like approach before performing feature fusion for tracking. Xin *et al.* [177] proposed an attention-based fusion of target and search features in a siamese backbone for transformer based tracking. Fan *et al.* [175] used correlation between the target feature and the current frame to localize the object. Transformers were proposed for distractor awareness by Wang *et al.* [176]. Mayer *et al.* [178] proposed a tracker in a correlation filter framework with bounding box regression. The ability to learn long-range dependencies along with self-attention as well as cross-

attention made them possible candidates for the next stage of evolution in tracking [20].

2.3.5 Learning methods

Different learning paradigms are used in training a deep neural network for visual object tracking. Reinforcement learning (RL) [133, 179–182], adversarial learning [183], few-shot learning and meta-learning are among the popular strategies explored. Reinforcement learning is used to decide the region to look for the given target in the frame [180]. Deep RL has two variants that are the Deep Q Networks (DQN) and policy gradient approaches. The DQN leans a state-action value function with the deep network. Policy gradient approaches learn policies on actions that maxmize reward. Yun *et al.* [133] was the first to incorporate RL in deep visual tracking. It was limited by its computational time. The object is located by a series of actions which are learned with deep reinforcement learning using policy gradient approach. Chen *et al.* [184] proposed a faster variant with Actor-Critic paradigm. Ren *et al.* [180] proposed an iterative approach to predict bounding box shift with reinforcement learning. This iterative approach is computationally less expensive when compared to sampling based approaches as less steps are required for convergence. Due to the iterative approach, this method was better at handling large deformation and motion. An actor-critic approach was followed to make decision on model update. Sun *et al.* [185] proposed a template update which automatically switches between multiple matching techniques with RL. Sun *et al.* [181] proposed a decision making policy which relied on tracking state history rather than on the current one. Zhang *et al.* [182] combined convolutional and recurrent neural networks with RL to leverage the use of spatio-temporal information.

Most deep learning trackers follow a tracing-by-detection approach [183]. Target samples that are collected from a frame have significant overlap among them and has a class imbalance compared to background samples. Song *et al.* [183] used adversarial learning [186] to capture object variations and addressed the class imbalance between target and background samples. More target samples are generated with adversarial leaning. Also, a cost sensitive loss function was designed to handle class imbalance. Meta-learning algorithm was applied with many tracking algorithms [10]. Anchor based detector were modified into tracker by Huang *et al.* [187] with a meta-learner [188]. The same was extended to anchor-free detectors by Wang *et al.* [189].

2.3.6 Ensemble Approach

Like classical approaches, ensemble techniques are also used with deep learning based algorithms. In their study, Wang *et al.* [32] highlighted the usefulness of ensemble in tracking. Diversity among the components is the key to the construction of a good ensemble. The first proposal for ensemble of deep motion features with appearance was made by Gladh *et al.* [80] in a tracking-by-detection framework. Deeper layers of a DNN extract more semantic features as compared to the shallower ones. Hierarchical features were used in [120, 126, 190]. Qi *et al.* [126] proposed combining correlation trackers built on embeddings from different layers of a DNN. Nam *et al.* [191] used multiple CNNs in a tree structure to have many appearance models to handle appearance changes of the object. Anfeng *et al.* [131] proposed a combination of two siamese networks. Each component focused on different features to bring diversity to the ensemble. One branch is trained to extract features relevant to similarity learning. The second branch is trained to extract semantic features that aids in classifications tasks. The output of each branch is a similarity map which is combined to locate the object. Teng *et al.* [192] proposed a combination of temporal and spatial networks. Luke *et al.* [193] proposed a single shot detector approach utilizing two models – one geometrically constrained in motion and the other invariant to transformation. Dunnhofer *et al.* [194] proposed a fusion of multiple trackers with the use of reinforcement learning.

More on deep learning based trackers can be found in [132, 195]. A snapshot of deep neural network based single object tracking literature over the years is presented in timeline 2.3.6. While developing an algorithm, data is required for training and testing a tracker. Image sequences with object ground-truth annotations are accessible in standard datasets available in public domain. Tracker proposals are evaluated on these datasets and compared using standard performance measures. To quantify algorithm performances, certain benchmarking tools are also available. These datasets, performance metrics and evaluation tools are described next.

TIMELINE 2: *Deep Neural Network based Trackers*

- 2015 Visual tracking with fully convolutional networks [196]
- 2016 Beyond correlation filters:Learning continuous convolution operators for visual tracking [156]
- 2016 Learning to track at 100 fps with deep regression networks [138]
- 2016 Fully-convolutional siamese networks for object tracking [10]
- 2016 Hedged Deep Tracking [126]
- 2016 Learning multi-domain convolutional neural networks for visual tracking [197]
- 2016 Siamese instance search for tracking [137]
- 2017 Action-decision networks for visual tracking with deep reinforcement learning [133]
- 2017 Eco: Efficient convolution operators for tracking [157]
- 2017 End-to-end representation learning for correlation filter based tracking [198]
- 2017 Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning [133]
- 2018 Distractor-aware siamese networks for visual object tracking [150]
- 2018 High performance visual tracking with siamese region proposal network [149]
- 2018 Vital: Visual tracking via adversarial learning [183]
- 2019 Learning the Model Update for Siamese Trackers [199]
- 2019 ATOM: Accurate Tracking by Overlap Maximization [160]
- 2019 Learning Discriminative Model Prediction for Tracking [161]
- 2019 Fast online object tracking and segmentation: A unifying approach [114]
- 2019 SiamRPN++: Evolution of Siamese Visual Tracking With Very Deep Networks [142]
- 2019 Target-Aware Deep Tracking [200]
- 2020 Siam R-CNN: Visual Tracking by Re-Detection [152]
- 2020 Probabilistic Regression for Visual Tracking [201]
- 2020 Siamese Box Adaptive Network for Visual Tracking [147]
- 2020 SiamCAR: Siamese Fully Convolutional Classification and Regression for Visual Tracking [146]
- 2020 Siam R-CNN: Visual tracking by re-detection [152]
- 2020 Know your surroundings: Exploiting scene information for object tracking [202]
- 2020 SiamAtt: Siamese attention network for visual tracking [203]
- 2021 Transformer tracking [177]
- 2021 Learn to match: Automatic matching network design for visual tracking [204]
- 2021 Learning Spatio-Temporal Transformer for Visual Tracking [205]
- 2021 Trtr: Visual tracking with transformer [206]
- 2021 Swintrack: A simple and strong baseline for transformer tracking [175]
- 2021 Do Different Tracking Tasks Require Different Appearance Models? [207]
- 2021 Transformer meets tracker: Exploiting temporal context for robust visual tracking [176]
- 2022 Robust Visual Tracking by Segmentation [208]
- 2022 Unified transformer tracker for object tracking [209]
- 2022 Global tracking via ensemble of local trackers [107]

.

2.4 Tracking Datasets

A tracking algorithm is initialized on the first image ($t = 0$) from a sequence of n^T images and predicts target object boundaries in the remaining images ($t = 1, \ldots n^T$). An image sequence dataset used for tracking also has target boundary annotations for each image in the sequence. For a good tracking algorithm, the predictions are expected to match the object ground-truth annotations. Table 2.2 presents a few datasets that are widely used for benchmarking tracking algorithms.

The target object boundaries are annotated in different ways. The object centroid, minimum bounding rectangle (upright or axis aligned), rotated minimum bounding rectangle and contour (realized as a polygon) are the most widely used ones as shown in Figure 2.18. Some datasets provide additional information on the scene regarding the challenges present in each frame. This helps in highlighting the performance of a tracker against certain challenges. For example Object Tracking Benchmark (OTB) dataset has information regarding illumination variation, scale changes, occlusion, shape deformations, motion blur and fast motion. These can be used to evaluate the performance of a tracker for specific challenging scenarios. Such an evaluation strategy is also seen with the Visual Object Tracking (VOT) benchmark. More on object tracking datasets can be found in [210].

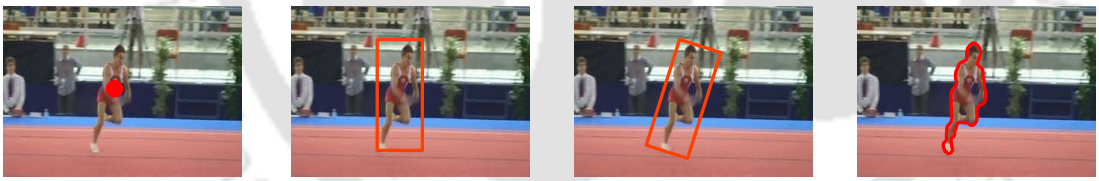

Figure 2.18: Illustrating the different ways of ground-truth annotation of target object. Object centroid, axis-aligned minimum bounding rectangle, rotated minimum bounding rectangle and object contour are the common ways of annotating target boundaries in datasets.

The datasets used for evaluation in this thesis include $VOT2018$, $VOT2019$, $OTB2015$ and $UAV123$. The $VOT2018$ and $VOT2019$ datasets have a total of 60 sequences. The groundtruth annotations are provided in the form of coordinates of rotated rectangles that cover the object. The $OTB2015$ has a total of 100 sequences and $UAV123$ has 123 sequences. The groundtruth annotations are in the form of upright (axis-aligned) bounding rectangles for $OTB2015$ and $UAV123$. These datasets have associated benchmarking tool which can evaluate tracker performance. $OTB2015$ dataset has more sequences as compared to $VOT2018$ dataset. Also, the challenges present in theses datasets are different as given in Table 2.1. UAV dataset images has a viewpoint change compared to VOT and OTB as

Name	Sequence Count	Benchmark Tool	Annotation	Attributes	Perf. Measures
$VOT2018$ [127]	60	Yes	Rotated Bounding Box	OCC, IV , OM, SZ, CM, Unassigned	Accuracy, Robustness, Expected Average Overlap
$VOT2019$ [211]	60	Yes	Rotated Bounding Box	OCC, IV , OM, SZ, CM, Unassigned	Accuracy, Robustness, Expected Average Overlap
$OTB2015$ [212]	100	Yes	Upright Bounding Box	IV, SV, OCC, DEF, MB,FM, IPR, OPR, OV, BC, LR	Success plot, Precision plot
$UAV123$ [213]	123	Yes	Upright (axis-aligned) Bounding Box	IV, SV, POC, FOC, OV,FM, CM,BC, SOB, ARC, VC, LR	Success plot, Precision plot

Table 2.1: Datasets used for performance evaluation. Attributes for the datasets include Illumination Variation (IV), Scale Variation (SV), Partial Occlusion (POC), Full Occlusion (FOC), Out of view (OV), Fast Motion (FM), Camera Motion (CM), Background Clutter (BC), Similar Object (SOB), Aspect Ratio Change (ARC), Viewpoint Change (VC), Low Resolution (LR), Occlusion (OCC), deformation (DEF), Motion Blur (MB), In Plane Rotation (IPR), Out of plane Rotation (OPR), Out of view (OV), Object Motion (OM), Object Size Change (SC)

these are aerial shots which has significant amount of shadows in it. Evaluation strategy followed and performance measures computed are different across VOT and OTB as provided by their respective toolkits.Table 2.1 presents a summary of the datasets used in this thesis. The performance metrics used for evaluating trackers are discussed next.

2.5 Performance Evaluation Measures

Different metrics are used to quantify the performance of a tracking algorithm from different perspectives [101, 214–216]. These measure also provide the means to compare multiple tracking algorithms. The measures may vary based on the target boundary annotation convention. Measures like accuracy, robustness, and expected average overlap are widely used to evaluate tracking algorithms. These measures reflect different aspects of the tracker performance. These measures are described next.

Centroid Error (Ψ^{CE}) [217] is computed as the Euclidean distance between the predicted and ground-truth position of the object bounding box centroids. For a single frame it is given as

$$\Psi_t^{CE} = \|a(\boldsymbol{\mathcal{I}}_t)^c - \mathcal{T}(\boldsymbol{\mathcal{I}}_t)^c\|^2 \tag{2.16}$$

where, $a(\boldsymbol{\mathcal{I}}_t)^c$ and $\mathcal{T}(\boldsymbol{\mathcal{I}}_t)^c$ are the respective bounding box centroids of groundtruth annotation and tracker prediction in $\boldsymbol{\mathcal{I}}_t$.

Precision Rate (Ψ^{PR}) is defined as the percentage of frames where the centroid error is lesser than a predefined threshold. Averaging Ψ_t^{CE} over the frames in a sequence provides the **Precision Rate** over the sequence.

$$\Psi^{PR} = \frac{1}{n^T} \sum_{t=1}^{n^T} \Psi_t^{CE} \mathbf{I}_{\Psi_t^{CE} \leq \lambda^{PR}} \tag{2.17}$$

Here, n^T is the total number of frames in the sequence and the indicator function $\mathbf{I}_{a \leq b}$ is defined as follows.

$$\mathbf{I}_{a \leq b} = \begin{cases} 1, & a \leq b \\ 0, & Otherwise \end{cases} \tag{2.18}$$

Different precision rates are computed for different threshold values ($\lambda^{PR} \in [0, 1]$) to generate the **Precision Plot** (Ψ^{PP}).

Intersection Over Union (IOU) (Ψ^{IOU}) evaluates the fractional overlap between the predicted target boundary and the ground truth annotation. The centroid error does not consider the scale of object. Hence, Ψ^{IOU} is considered the better one among the two. The Ψ_t^{IOU} for a frame $\boldsymbol{\mathcal{I}}_t$ is computed as follows

$$\Psi_t^{IOU} = \frac{|a(\boldsymbol{\mathcal{I}}_t) \cap \mathcal{T}(\boldsymbol{\mathcal{I}}_t)|}{|a(\boldsymbol{\mathcal{I}}_t) \cup \mathcal{T}(\boldsymbol{\mathcal{I}}_t)|} \tag{2.19}$$

where, $a(\boldsymbol{\mathcal{I}}_t)$ and $\mathcal{T}(\boldsymbol{\mathcal{I}}_t)$ are the respective ground truth annotation and tracker prediction of target boundary. Averaging over the total number of frames n_T provides the performance measure Ψ^{IOU}. This measure is also referred to as **Accuracy (A)**.

$$\Psi^{IOU} = \frac{1}{n^T} \sum_{t=1}^{n^T} \Psi_t^{IOU} \tag{2.20}$$

Success Rate (Ψ^{SR}) is defined as the percentage of frames where the Ψ_t^{IOU} exceeds a predefined threshold. It is also referred as **Accuracy (A)**. Different success rates are obtained by varying the threshold $\lambda^{SR} \in [0, 1]$. These success rates are used to generate the **Success Plot**.

Name	Datasets	Bench marking Tool
ALOV [1]	ALOV300	-
DTB70 [218]	DTB70	-
GOT-10k [219]	GOT-10K	Yes
LaSOT [220]	LaSOT	Yes
LTW [221]	LTW	-
Need for speed [222]	Need for speed	-
OTB [212]	OTB50,OTB2015	Yes
TrackingNet [223]	TrackingNet	-
Temple-Color [224]	TempleColor128	-
UAV [213]	UAV123, UAV20L	-
VOT [101]	2013-2021	Yes

Table 2.2: Popular Tracking datasets and benchmarks

$$\Psi^{SR} = \frac{1}{n^T} \sum_{t=1}^{n^T} \Psi_t^{IOU} \mathbf{I}_{\lambda^{SR} \leq \Psi_t^{IOU}} \tag{2.21}$$

The quantities $\Psi_{0.50}^{SR}$ and $\Psi_{0.75}^{SR}$ respectively represent the success rates at overlap threshold values of $\lambda^{SR} = 0.5$ and $\lambda^{PR} = 0.75$. Ideally, a tracker has to accurately localize the target object in all frames of the input sequence. A tracking failure is detected when Ψ^{IOU} or Ψ^{CE} of a frame drops below a predefined threshold. The **Track Length** [214] is the number of frames tracked between initialization and first failure. This is also referred as the longevity of the tracker.

Robustness (R) (Ψ^R) [214] follows a supervised technique where the tracker is re-initialized on failure. Failure is identified in a frame when $\Psi_t^{IOU} \leq \lambda^R$. Here, $\lambda^R \in [0, 1]$ is the parameter that defines when to detect failure. The average number of reinitializations required by a tracker for all input sequences is defined as the robustness of the tracker for the dataset. Note that a lower value of robustness indicates better tracking performance. Kristen *et al.* [101] showed that the measures accuracy and robustness complement each other. Robustness of a sequence is computed as

$$\Psi^R = \frac{1}{n^T} \sum_{t=1}^{n^T} \mathbf{I}_{\Psi_t^{IOU} \leq \lambda^R} \tag{2.22}$$

Expected Average overlap (EAO) [214] is the average IOU on smaller sequences of varying length derived from the larger sequences.

2.6 Evaluation Methodology

The trackers proposed in this thesis are evaluated to quantify their performance against baseline algorithms. In this thesis, the GOT-10K and the VOT toolkits are used for this evaluation. The $VOT2019$, $OTB2015$ and $UAV13$ datasets are evaluated with GOT-10K. Performance on $VOT2018$ dataset is evaluated with VOT toolkit. Performance of other tracking algorithms are available with the toolkit for comparison. These toolkits provide an independent platform to evaluate and compare proposed trackers with the baselines. Some of the commonly used datasets and benchmarking tool are listed in table 2.2. The GOT-10K toolkit is used for evaluating performance on datasets $VOT2019$, $OTB2015$ and $UAV123$.

One pass evaluation (OPE) [212] strategy is followed in $OTB2015$ and $UAV123$ datasets where the tracker is initalized on the first frame and is allowed to run till the end of the sequence without any re-initialization. Success plot and precision plot are used for evaluation. The measures Ψ^{CE} and Ψ^{IOU} are computed over the datasets. The success and precision plots of the baseline trackers available on $OTB2015$ dataset is given in Figures A.1 and A.2 respectively.

The VOT toolkit re-initializes a tracker upon failure ($\Psi_t^{IOU} \leq \lambda^R$). The performance measures used in the VOT toolkit include Accuracy, Robustness and Expected Average Overlap. For VOT evaluation, tracker is initialized with the bounding box $\mathbf{bb}_0^+$ in first frame and is then re-initialized after skipping 5 frames upon failure. Annotations are in the form of upright bounding box and oriented bounding box depending on the dataset versions. After initialization, the object is tracked in the subsequent frames and Ψ_t^{IOU} is computed. Whenever a failure happens, 5 frames are skipped and the tracker is re-initialized to continue tracking. The overall accuracy is defined as the average Ψ^{IOU} on all sequences in the dataset. The overall robustness is the average number of failures made by the tracker over all sequences in a dataset. Benchmark results of trackers on $VOT2018$ is available in appendix A.1 for further reference. Better performance of a tracker is indicated by higher values ($\uparrow$) of accuracy, expected average overlap, success rate, precision rate, and lower value ($\downarrow$) of robustness.

2.7 Evaluated Algorithms

For performance comparison, benchmark trackers available with the VOT and OTB are used. Algorithms belonging to the groups of both classical approaches as well as deep learning based techniques are available for comparison. To keep things fair, the proposed classical approaches(Chapters 3

and 4) are compared with algorithms from classical approach group and the modern ones (Chapter 5) are compared with the contemporary literature. Some of the algorithms that are selected for comparison from the benchmarks are given below.

VOT2018: L1APG [9], Matflow [127], BDF [127], BST [127], DSST [62], FoT [127], IVT [225], KCF [60], MIL [217], STRUCK [76], FragTrack [102], LGT [127], Matrioska [127]

VOT2018: DSiam [157], ECO [157], RAnet [211] , SiamFC [10], SiamRPN [144], SiamVGG [211] , SRDCF [211], Staple [69] , UpdateNet [199]

VOT2019: DPT [211], RankingT [211] , RSiamFC [211] , SiamFCOSP [211], SiamMask [149], SiamRPNX [211] , TADT [200], TDE [211]

2.8 Discussions

Visual object tracking continues to be a challenging research problem in the domain of computer vision. Extensive research highlights that no single tracker can handle all challenging scenarios. It is observed that a successful tracker should have the following components. First, the use of hand-crafted or learned features that are robust to illumination changes, shape deformations, occlusions and background clutter. Second, an object model (constructed with these features) with good representation capability. Third, a discriminative model (constructed with these features) for identifying target against background and distractors. Fourth, an effective scheme for updating the models for enabling long-term tracking. Fifth, fast search strategies for object localization. Finally, tracker ensembles can be explored for handling a variety of challenging scenarios.

Image pixel intensity (color) distribution is one of the simplest features. It is also capable of accommodating minor appearance variations, in-plane rotations and scale changes [4]. Color distributions from both (potential) target and (immediate) background regions are explored in Chapter 3. An objective function defined over these distributions is optimized using meta-heuristic search strategies for object localization.

Color distribution learned from the entire object ignores local structural information. Instead of using a single global color model, the object is divided into parts and local color models are learned on them. This multi-pat model approach is explored in the context of occlusion handling as well. Adam *et al.* [102] showed improved performance with multi-part histogram based approach. Part-based models do help in handling occlusions as well [33]. Semantic information of the object is used for

object modeling in literature. Contextual information among object pixels is retained with a template based model which is lost in the color distribution based model. An object template is less robust to appearance change due to the presence of more semantic information in it. Multiple templates are used to address this. However, a better approach is the use of object patches and multiple variants of the same to capture object appearance changes. This allows the model to capture object deformations. The same can be said about deep neural network features. Relatively invariant features are observed at shallow layers of a DNN. Deeper ones have more semantic information but are not invariant to rotation and deformations. Invariance can be improved with the use of object patches rather than the full template. In this context, sparse representation of patch vectors is explored in Chapter 4 for object modeling. The transformed vectors should be able to accommodate minor variations in object patches. As seen with the color based models, sparse representation alone provide robustness against distractors and occlusion. To improve discrimination, background information as well as use of classifiers is explored in creating an object model. A multi-part approach is used to handle occlusion. Sparse representation uses learning from data approach but is limited by the amount of training data available. Only information from the first frame of a sequence is available to model the object.

The two important characteristics of an object model are its discrimination power and its generalization capability. Discrimination helps to differentiate object from the background. A generalizable model can accommodate object appearance changes. Color distribution is a generalizable model but is not that discriminant when compared with semantic features. A pre-trained deep neural network embedding can provide a balanced feature set as the features are learned from a larger multi-class dataset in a supervised manner. Also, the feature embeddings of a class is learned from the different appearances present in the training data. These two properties make a deep neural network a good candidate for feature extraction. Semantic features learned from data are used for representing the object in a deep neural network based tracker. Localization is performed with the convolutional operation by comparing embeddings of the object with that of a search image. This similarity learning is done in a siamese architecture framework.

Additional cues in the form of motion is used to re-rank predictions from the appearance based trackers. Motion information of object and distractor might be different and therefore can be used to differentiate them. Motion is considered complementary to appearance in tracking literature. It is widely explored and is shown that their combination can bring performance improvement [69].

In this thesis, motion is used as an additional cue to re-rank predictions from the appearance based trackers (Chapters 3, 4, and 5) to improve tracking performance.

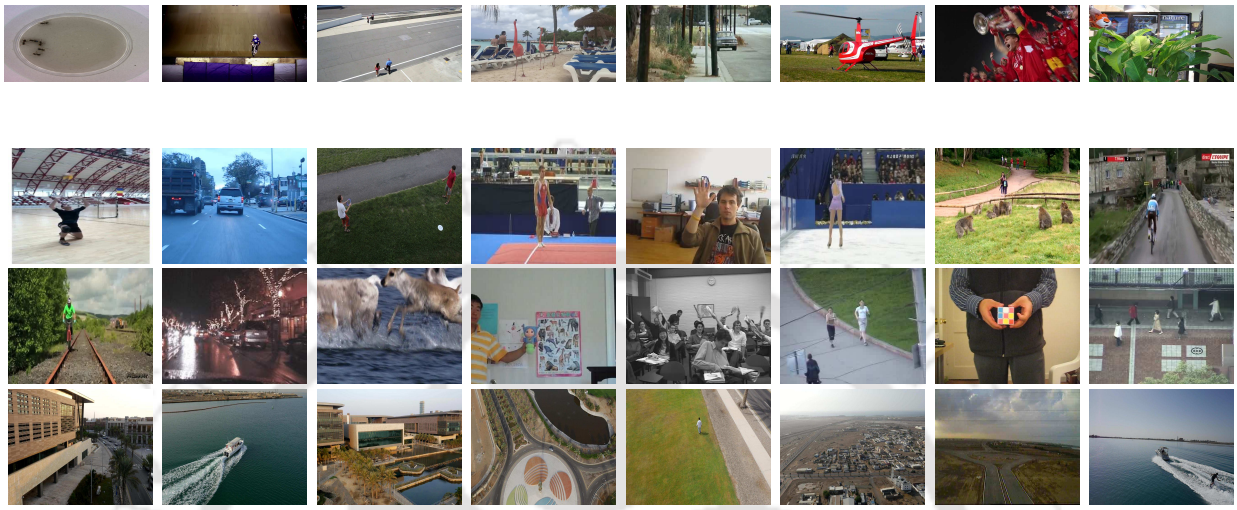

Figure 2.19: Each row shows frames from sequences in $VOT2018$, $VOT2019$, $OTB2015$ and $UAV123$ datasets respectively.

3

Color Model based Tracking

Publications

- Pranay Kate, **Mathew Francis**, Prithwijit Guha, "Visual tracking with breeding fireflies using brightness from background-foreground information", in *International Conference on Pattern Recognition (ICPR)*, 2018.

Contents

Overview

This work explores color as a feature for constructing an object model. Distribution of color pixel intensities in the target region is used for modeling. Such a representation is robust to geometric transformations like scaling and in-plane rotation. The target and background color distributions are expected to be different in a sequence. However, this assumption is often violated due to rectangular box based initial annotation of non-rectangular target objects, background clutter and presence of other similar objects (distractors). This often leads to tracking failures. Thus, this work emphasizes features specific to target object by using the context information from background region around the object. This object model is called the "Context Weighted Color Distribution" (CWCD) and is used for object localization. Several localization schemes involving probabilistic, evolutionary and swarm based approaches are explored. A proposal is made to combine Firefly Algorithm (FA, a swarming technique) with Real coded Genetic Algorithm (RGA, an evolutionary approach) for better target localization. Complementary information from inter-frame motion is used to improve the tracker prediction with a re-ranking approach. Evaluations are performed on $VOT2018$, $OTB2015$ and $UAV123$ datasets.

3.1 Color Model based Trackers: An Introduction

Object modeling and localization are essential building blocks of a tracking algorithm. The object models are often constructed as statistical models learned from the object appearance features [19]. In this chapter, a pixel color intensity based generative object model is explored. Such a model can accommodate small intensity variations that an object might undergo in a sequence. Also, it is robust to object scaling and in-plane rotations. The mean-shift tracking algorithm proposed by Comaniciu *et al.* [110] has demonstrated the efficacy of color distribution. This feature is widely explored in tracking literature [2] owing to its simplicity and effectiveness. Several other descriptors and models were proposed in the literature to accomplish tracking. Some popular descriptors include Scale Invariant Feature Transform (SIFT) [28], Oriented and Rotated BRIEF (ORB) [29], Fast Retina Keypoint (FREAK) [30], sparse codes [31] among others. Other features include object template [33], Histogram of Oriented Gradients [31], learned detectors [35] etc..

During object localization, candidate models are constructed from image region proposals that are generated from the object state space. An objective function based on similarity (or dissimilarity) between the target and candidate model is maximized (or minimized) to identify the best region proposal. The object is localized with this optimal image region. This work uses color distribution as object model. Object localization is performed by maximizing (or minimizing) the similarity (or dissimilarity) between the target and candidate color distributions. Different color distribution similarity [110, 226] or dissimilarity [227, 228] metrics have been used in the literature.

Measures like correlation [22], reconstruction error [9, 229], cosine similarity [230], Bhattacharya

coefficient [110], histogram intersection [226], mover's distance (EMD) [227], Kullback–Leibler divergence [228], object or background likelihood values [43] are used among others in literature. Among these, the Bhattacharyya coefficient is widely used to compute similarity between distributions [102, 110, 231–233].

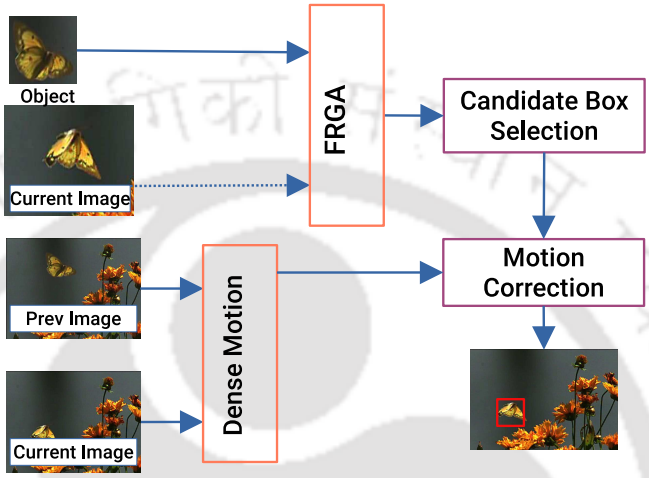

Figure 3.1: The functional block diagram of the ensemble tracker. Target foreground and background models are computed from frame $\mathcal{I}_0$. Object is identified in the subsequent frames using *FRGA* localization algorithm.

Different localization algorithms exist in literature that maximize (or minimize) a certain similarity (or dissimilarity) function in tracking. Brute force search for target within a search region is often applied in template based matching [22]. Here, the object is localized by maximizing the normalized cross-correlation score between target and candidate templates. Comaniciu *et al.* [110] introduced an iterative search algorithm for minimizing the dissimilarity (Bhattacharya distance) between target and candidate color distributions. These two were deterministic search strategies. However, object localization is mostly dominated by randomized search techniques like Particle Filter (PF) and its variants [234, 235]. These methods have dominated the tracking literature due to their simplicity and the ability to provide localization where gradient based search is hard (e.g. in scale or orientation space).

The object is localized in its state space $[x, y, w, h, \theta]$ during tracking. The search for the optimal solution often depends on the dimension of the object state vector. Meta-heuristic search algorithms were found to provide faster convergence (to optima) in high dimensional search spaces [24]. These algorithms generate solutions by following a set of rules in a random yet directed manner [24]. This directed approach leads to better convergence and the randomness helps in search space exploration and exploitation [24]. Such approaches can be broadly divided into two categories. First, the evolu-

tionary search strategies where a solution set is treated as a population of entities. These entities (or solutions) interact and breed across generations (or iterations) thereby producing an evolved population (or better solution set). Researchers have proposed different methods for *solution representation*, *survival of the fittest solution*, *parent solution selection for breeding* and *genetic operations* [24]. This led to different evolutionary search strategies like Binary coded Genetic Algorithm [90], Differential Evolution (DE) [236], Real Coded Genetic Algorithm (RGA) [237] and their variants. The second category represents the solution set as a population of particles (moving around) in the search space. A set of predefined rules emulate the swarming behaviour of the particles by deciding their position and motion parameters. This drives the entire particle population towards the optima in the search space. However, in swarm based techniques, the particles do not breed. Notable swarm based search techniques include Particle Swarm Optimization (PSO) [85], Firefly Algorithm (FA) [86], Ant Colony Optimization (ACO) Algorithm [238], Grey Wolf Algorithm [239], Cuckoo Search [240] among others.

Evolutionary algorithms are based on generation update and swarm approaches are based on position update [241]. In evolutionary approaches, the best solutions in an generation are selected for creating the next one. On the other hand, the absence of a selection criteria can make swarm based techniques less efficient in reaching the optimal solution [242]. Inter-particle interactions within the swarm enhances local search for optimal solution while evolutionary approaches are good at global search [7, 242]. Thus, combining these two approaches can yield an algorithm with diverse solutions having the capability to reach the optima [242]. Here, the evolutionary component promotes global search and swarm component enhances local search [242].

Such hybrid approaches combine the position and velocity update of particles (in swam) approach with genetic operators from evolutionary techniques [242]. For example, breeding swarm strategies [242, 243] were proposed by combining Particle Swarm Optimization (PSO) with Real-coded Genetic Algorithm (RGA) [237].

Firefly Algorithm (FA) was developed based on the flashing pattern of tropical fireflies. In FA, the population can divide themselves into groups around different local optima [86]. This helps in the search for the global optima [86]. This makes it a suitable method for handling objective functions with multiple modes. A combination of binary coded genetic algorithm (evolutionary search strategy) and FA (swarming technique) is proposed as Firefly Mating Algorithm (FMA) [244]. Gao *et al.* [26]

have proposed the use of FA in the localization stage of tracking. A modification is proposed in the firefly position update stage while considering the target movement [25]. An update of particle positions using FA prior to resampling in particle filter based tracking was proposed in [81]. A complete discussion on FA is out of the scope of this chapter. More on the algorithm and its applications can be found in [228].

Several existing approaches have used motion cues in object tracking. Inter-frame dense motion can be estimated using Horn–Schunck [89] or Farenback [91] optical flow methods. Motion estimation between frames $\mathcal{I}_{t-1}$ and $\mathcal{I}_t$ can locate object pixels in $\mathcal{I}_t$ to accomplish object tracking. Several tracking algorithms are proposed by using inter-frame motion [74, 245]. Mae *et al.* [34] proposed a combination of edge information and optical flow to perform tracking in cluttered environments. SIFT feature correspondences are combined with optical flow information in [246]. These motivated this work to combine both appearance and motion cues in tracking.

A color intensity based object model can get corrupted in two ways. First, there is a possibility that some of the colors might be common to both object and background. Second, the approximation of object boundary during target region annotation in first frame may introduce immediate background. For example, a rectangular bounding box (most commonly used) for a non-rectangular object will always include some portion of the immediate background. This will result in some background information within the object model when the tracker is initialized. Many proposals have been made in the literature to overcome the effect of background. Comaniciu *et al.* [110] proposed the use of Epanechnikov kernel weighted color distribution construction. In this approach, the pixel weight increases from the periphery to the centroid of object bounding ellipse (support of kernel function). However, this approach does not check whether a particular pixel intensity is important for the object model or not. To this end, Ning *et al.* [40] proposed a background weighted histogram where unique colors of the object are emphasised. A similar approach is adopted in this work where object and its immediate background or *context* is explicitly modeled. The object model is refined with background information to emphasize object-only-colors. This improves the discrimination capability of the model. Apart form using the contextual information in refining the object model, it is explicitly used in defining an objective function for the localization step. A potential candidate region should have **(a)** maximum similarity to an object and **(b)** minimum similarity to the object background. In this chapter, a new objective function is defined to realize this criteria. It is computed as a combination

of Bhattacharyya similarity of a candidate model with target and Bhattacharyya dissimilarity of target model with candidate background. This is then used in the localization stage of tracking.

The Firefly Algorithm (FA) is a swarming technique and does not involve particle breeding. A breeding swarm based approach enhances search performance as compared to either only evolutionary strategy or swarm based technique [7]. The breeding component aids in reaching near the optimal solution [242]. The Firefly Algorithm (FA) was found to be better among several swarm based approaches as concluded in [86]. In this work, modifications on the basic FA is achieved with a breeding swarm strategy to generate diverse solutions. The breeding component is realized with Real Coded Genetic Algorithm (RGA). The modified approach is called Firefly with Real-coded genetic algorithm (FRGA). Firefly brightness values are defined from the proposed objective function that utilizes contextual information in addition to object information.

Santner *et al.* [77] showed that motion is considered complementary to appearance feature and their combination can improve overall tracking performance. In this work, motion information is used to re-rank the tracking predictions obtained from the localization algorithm. The FRGA algorithm works with the proposed context weighted color distribution (appearance feature). In this work, a combination of FRGA with motion prediction is used in the localization stage. Predictions from the FRGA component is refined with the inter-frame motion vectors to select the best candidate region proposal.

Major contributions of this chapter are summarized as follows.

(i) A novel Context Weighted Color Distribution (CWCD) based object model

(ii) Novel objective function that includes contextual information

(iii) Modified Firefly algorithm with RGA component

(iv) Motion based re-ranking in ensemble tracker

Extensive experiments to highlight the advantages of the novel objective function and to compare localizations schemes are performed on $VOT2018$, $OTB2015$ and $UAV123$ datasets.

Rest of the chapter is organised as follows. Section 3.2 explains the proposed object model. The various localization schemes are described in Section 3.3. The experiments and results are presented in Section 3.5. Finally, the chapter is summarized in Section 3.6 with discussions on possible future extensions.

3.2 Object Model

A tracker with a color distribution based object model is explored in this chapter. It is constructed during the initialization phase. A target object is identified by its (minimum) bounding rectangle $\mathbf{bb}_0^+$ in first frame $\mathcal{I}_0$ of a sequence. A larger concentric rectangle $\mathbf{bb}_0^\pm$ that contains contextual information as well is defined to capture the immediate background information. A region consisting of only background is further defined as

$$\mathbf{bb}_0^- = \mathbf{bb}_0^\pm - \mathbf{bb}_0^+ \tag{3.1}$$

Color distribution is used for modeling both target (initial frame) and candidates (later frames for localization). The model is constructed from object and its immediate background (context). The target model is refined with the help of contextual information thereby, making it more robust. Color being a simple feature, its distribution can be easily computed from the pixel intensity values. Binning of colors in a histogram allows room for accommodating minor ,intensity variations which helps in handling minor intensity variations. Also, color distribution is robust to scaling and in-plane rotation of the object.

3.2.1 Object Color Distribution

Figure 3.2: Color distribution of object computed from object bounding box $\mathbf{bb}_0$

.

A target object is initially identified by its bounding box $\mathbf{bb}_0^+$ as an image region $\mathcal{I}^Q$ in the initial image $\mathcal{I}_0$. Here a statistical model with color pixel intensity distribution is used. The tracking algorithm constructs target object model $\mathbf{m}_0^{\mathbb{T}+} = \mathbf{g}(\mathcal{I}_0, \mathbf{bb}_0^+)$ using feature distribution computed from the initial image $\mathcal{I}^Q \in \mathcal{I}_0$. Model generation is shown in figure 3.2. Function $\mathbf{g}(\mathcal{I}, \mathbf{bb}^+)$ computes a model from the region proposal $\mathbf{bb}^+$ and frame $\mathcal{I}$. The target model ($\mathbf{m}_0^{\mathbb{T}+}$) is computed as follows.

$$\mathbf{m}_0^{\mathbb{T}+}[j] = L^+ \sum_{i=1}^{n_+} k(\|\mathbf{x}_i^\star\|^2)\delta[b(\mathbf{x}_i^\star) - j] \tag{3.2}$$

Here, $\delta[\cdot]$ is the Kronecker delta function and L^+ is the normalization constant. During tracking, region proposals are generated and a similar process is applied to construct candidate models $\mathbf{m}_t^{\mathbb{C}} = \mathbf{g}(\boldsymbol{\mathcal{I}}_t, \mathbf{bb}_t^+)$ for the proposals. Consider a search image $\boldsymbol{\mathcal{I}}_t^S \in \boldsymbol{\mathcal{I}}_t$. Candidate region proposals are generated within $\boldsymbol{\mathcal{I}}_t^S$ to form the set $\mathbb{B}_t^+ = \{\mathbf{bb}_{ti}^+, i = 1 \ldots, n_{\mathbb{B}}\}$. Concentric bounding rectangle set $\mathbb{B}_t^\pm$ and $\mathbb{B}_t^-$ are generated as follows

$$\mathbf{bb}_{ti}^- = \mathbf{bb}_{ti}^\pm - \mathbf{bb}_{ti}^+ \tag{3.3}$$

Here, $\mathbf{bb}_{ti}^\pm \in \mathbb{B}_t^\pm$ and $\mathbf{bb}_{ti}^- \in \mathbb{B}_t^-$.

Color pixel intensity feature is widely used in many computer vision tasks [247–249]. However, for object tracking it is also important to differentiate target from the background. A model solely based on object color distribution would not suffice in this case. A modification is proposed next.

3.2.2 Context weighted Object Color Distribution

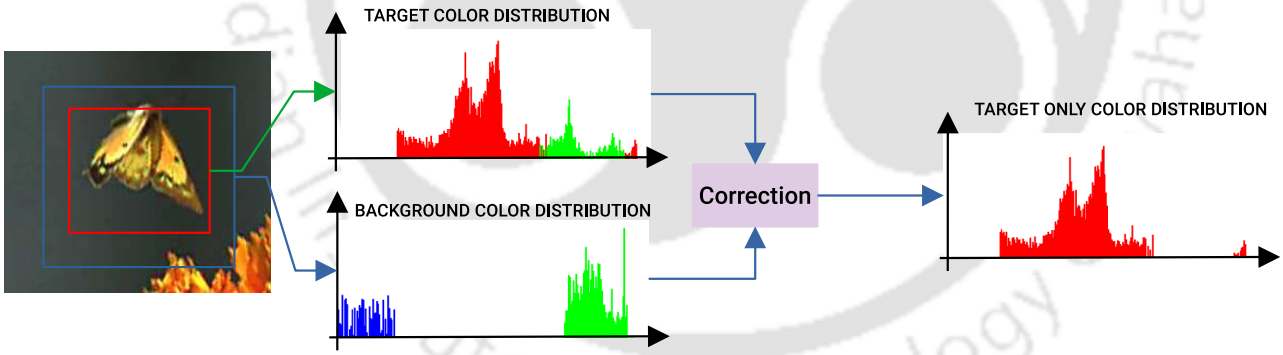

Figure 3.3: **(a)** Color distribution of object (red) and background (blue). Common features, highlighted in green are removed from initial target color distribution to obtain the final target model (color distribution).

The object bounding box $\mathbf{bb}_0^+$ might include the information of immediate background. This usually happens with rectangular bounding box annotation of a non-rectangular target object. Also, background clutter and object with similar color (distractors) might result in overlapping information between $\mathbf{bb}_0^+$ and $\mathbf{bb}_0^-$. In this proposal, the target background is explicitly modeled for obtaining contextual information as $\mathbf{m}_0^{\mathbb{T}-} = g(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_0^-)$ along with the target model. Thus the target model has a foreground and background component $\mathbf{m}_0^{\mathbb{T}} = \{\mathbf{m}_0^{\mathbb{T}+}, \mathbf{m}_0^{\mathbb{T}-}\}$ computed as follows

$$\mathbf{m}_0^{\mathbb{T}+}[j] = L_{\mathbb{T}}^{+} \sum_{i=1}^{n_+} \delta[b(\mathbf{x}_i^\star) - j] \tag{3.4}$$

$$\mathbf{m}_0^{\mathbb{T}-}[j] = L_{\mathbb{T}}^{-} \sum_{i=1}^{n_-} \delta[b(\mathbf{x}_i^\star) - j] \tag{3.5}$$

where, $j = 1, \ldots, n_b$, $b(\mathbf{x})$ computes the color bin of the pixel at image position $\mathbf{x}$, $\delta[\cdot]$ is the Kronecker delta function and $L_{\mathbb{T}}^{+}$, $L_{\mathbb{T}}^{-}$ are normalization constants, n_+, n_- are the respective number of pixels in $\mathbf{bb}_0^{+}$ and $\mathbf{bb}_0^{-}$.

Background information in target model is suppressed by identifying colors that are common to both object and background. The modified model $\mathbf{m}_0^{\mathbb{T}+}$ is computed from equation (3.6). This is used as the final target model for object localization. Let $\mathbf{J}^c = \{j_i^c, i = 1, \ldots, n_{bc}\}$ be the set of n_{bc} bins that are common to both object and background such that $\mathbf{m}_0^{\mathbb{T}-}[j] \geq \mathbf{m}_0^{\mathbb{T}+}[j]$. Modified target foreground model is defined as follows.

$$\mathbf{m}_0^{\mathbb{T}+}[j_i^c] = 0.0, \forall\ j\ s.t\ \mathbf{m}_0^{\mathbb{T}-}[j_i^c] \geq \mathbf{m}_0^{\mathbb{T}+}[j_i^c] \tag{3.6}$$

In t^{th} frame $\boldsymbol{\mathcal{I}}_t$, a similar process is applied to construct candidate model $\mathbf{m}_t^{\mathbb{C}} = \mathbf{g}(\boldsymbol{\mathcal{I}}_t, \mathbf{bb}_t^{+})$ from region proposal $\mathbf{bb}_t^{\pm}$. The candidate models are computed as follows

$$\mathbf{m}_t^{\mathbb{C}+}[j](\mathbf{x}) = L_C^{+} \sum_{i=1}^{n_+} \delta[b(\mathbf{x}_i) - j] \tag{3.7}$$

$$\mathbf{m}_t^{\mathbb{C}-}[j](\mathbf{x}) = L_C^{-} \sum_{i=1}^{n_-} \delta[b(\mathbf{x}_i) - j] \tag{3.8}$$

where, $\mathbf{x}$ is the centroid of $\mathbf{bb}_{ti}^{+}$ as well as $\mathbf{bb}_{ti}^{-}$, L_C^{+}, L_C^{-} are normalization constants, n_+, n_- are the respective number of pixels in $\mathbf{bb}_{ti}^{+}$ and $\mathbf{bb}_{ti}^{-}$.

The target and candidate models are used in the localization stage to identify the best candidate solution. The localization algorithm aims to optimize an objective function for identifying target region in the given frame $\boldsymbol{\mathcal{I}}_t$. The proposed objective function is defined by using both target and candidate models and is explained next.

3.2.3 Objective Function

During tracking, candidate proposals are generated and corresponding models are also computed. Localization in performed in the position (x, y), scale (w, h) and orientation θ. Candidate bound-

Figure 3.4: Image on the left shows an optimal box and its contextual box. In the center the estimated box is a suboptimal one whose dimensions are less than the optimal box (scale implosion). Image on the right shows an estimated box whose dimensions are large in comparison with the optimal box(scale explosion).

.

ing box proposals are uniformly sampled from a smaller region defined around the prediction from previous frame ($\overset{\star}{\mathbf{bb}}{}^{+}_{t-1}$). This small region is referred as the search image $\boldsymbol{\mathcal{I}}^S \in \boldsymbol{\mathcal{I}}_t$.

Let the set of all such bounding boxes be $\mathbb{B}^+_t = \{\mathbf{bb}^+_{ti}, i = 1, \ldots, n_{\mathbb{B}}\}$. Similarity computation is essential to identify the optimal bounding box $\overset{\star}{\mathbf{bb}}{}^{+}_{t}$ from the candidate proposal set $\mathbb{B}^+_t$. The Bhattacharyya similarity measure [110] $\rho(\mathbf{m}_0^{\mathbb{T}+}, \mathbf{m}_t^{\mathbb{C}+}) \in [0, 1]$ has been widely used in the literature for comparing target and candidate models and is given as follows.

$$\rho^+_{ti}(\mathbf{m}_0^{\mathbb{T}+}, \mathbf{m}_{ti}^{\mathbb{C}+}) = \sum_{j=1}^{n_b} \sqrt{\mathbf{m}_0^{\mathbb{T}+}[j] \times \mathbf{m}_{ti}^{\mathbb{C}+}[j]} \tag{3.9}$$

An object is usually different from its background. However, this property is not captured in equation 3.9. This can be measured by computing how dissimilar the candidate background is from the object model. The Bhattacharyya similarity between $\mathbf{m}_0^{\mathbb{T}+}$ and $\mathbf{m}_t^{\mathbb{C}-}$ it can be defined as follows.

$$\rho^-_{ti}(\mathbf{m}_0^{\mathbb{T}+}, \mathbf{m}_{ti}^{\mathbb{C}-}) = \sum_{j=1}^{n_b} \sqrt{\mathbf{m}_0^{\mathbb{T}+}[j] \times \mathbf{m}_{ti}^{\mathbb{C}-}[j]} \tag{3.10}$$

Equation 3.9 computes how similar a target foreground model is to a candidate foreground model. Similarity with the candidate background model and target foreground model is computed with equation 3.10. For a good canidate model ρ^+_{ti} should be high and ρ^-_{ti} should be low. Such a candidate box $\mathbf{bb}^{\pm}_{ti}$ will have maximum target information in $\mathbf{bb}^+_{ti}$ and minimum target information in $\mathbf{bb}^-_{ti}$.

The optimal bounding box $\overset{\star}{\mathbf{bb}}{}^{+}_{t}$ is selected from the regional proposals during the localization phase. This optimal box is the one that draws a minimum bounding rectangle while enclosing the tracked object. A region proposal $\mathbf{bb}^{\pm}_{ti}$ has both foreground $\mathbf{bb}^+_{ti}$ and background $\mathbf{bb}^-_{ti}$. Most optimum candidate box $\mathbf{bb}^{\pm}_{ti}$ will have a foreground box $\mathbf{bb}^+_{ti}$ with object in it and a background region $\mathbf{bb}^-_{ti}$ with background in it. Object presence in $\mathbf{bb}^+_{ti}$ is measured with equation 3.9 and background presence

in $\mathbf{bb}_{ti}^{-}$ using equation 3.10. In the measure ρ_{ti}^{-} background is assumed as anything but object. An incorrect region proposal will have object and(or) background in both $\mathbf{bb}_{ti}^{+}$ and $\mathbf{bb}_{ti}^{-}$ resulting in a lower value for both ρ_{ti}^{+} and ρ_{ti}^{-}. For example, consider a region proposal on the object with a smaller size than $\overset{\star}{\mathbf{bb}}_{t}^{+}$. It will have only object in $\mathbf{bb}_{ti}^{+}$ but both object and background in $\mathbf{bb}_{ti}^{-}$. This will still result in higher value from ρ_{ti}^{+} but a very low value from ρ_{ti}^{-} as the object pixels are present in $\mathbf{bb}_{ti}^{-}$. Similarly a larger region proposal results in the presence of background in $\mathbf{bb}_{ti}^{+}$ resulting in a lower value from first component of ρ_{ti}^{+}. The same reasoning can be applied to incorrectly positioned or oriented region proposals. Thus a combination of ρ_{ti}^{+} and ρ_{ti}^{-} will promote the selection of a box with a tighter boundary around the object. Thus, the proposed combined measure $\rho_{ti}^{\pm}$ is given as follows.

$$\rho_{ti}^{\pm}(\mathbf{m}_{0}^{\mathbb{T}+}, \mathbf{m}_{ti}^{\mathbb{C}+}) = \rho_{ti}^{+}(\mathbf{m}_{0}^{\mathbb{T}+}, \mathbf{m}_{ti}^{\mathbb{C}+}) \left(1 - \sum_{j=1}^{n_b} \sqrt{\mathbf{m}_{0}^{\mathbb{T}+}[j] \times \mathbf{m}_{ti}^{\mathbb{C}-}[j]}\right) \tag{3.11}$$

Here, $\rho_{ti}^{\pm}$ promotes the selection of a region (from the proposals in $\mathbb{B}_{t}^{+}$) which has maximum object information in $\mathbf{bb}_{ti}^{+}$ and minimum object information in $\mathbf{bb}_{ti}^{-}$. This means ρ_{ti}^{+} fails in cases of scale implosion where the algorithm progressively selects smaller regions leading to an eventual collapse. The combined measure $\rho_{ti}^{\pm}$ helps overcome this, as a smaller box will reduce ρ_{ti}^{-} leading to a reduction in overall score. The combined score helps to obtain a tighter boundary around the object resulting in better localization during tracking. The problem of scale implosion and explosion are illustrated in Figure 3.4. The target model $\mathbf{m}_{0}^{\mathbb{T}+}$ is updated to capture the appearance changes an object might undergo as explained next.

3.2.4 Model Update

Model update is essential to capture the gradual, variations of object appearance as tracking progresses. Too frequent update can corrupt the model and cause tracker drift. Similarly, no update might result in gradual track loss. To achieve long term tracking appearance changes are to be incorporated into the model. Update is done as follows.

$$\mathbf{m}_{t}^{\mathbb{T}+} = \alpha_u \mathbf{m}_{t-1}^{\mathbb{T}+} + (1 - \alpha_u)\mathbf{m}_{ti^\star}^{\mathbb{C}+} \tag{3.12}$$

Here, $\mathbf{m}_{ti^\star}^{\mathbb{C}+}$ is the candidate model for the predicted box $\overset{\star}{\mathbf{bb}}_{t}^{+}$ in frame $\boldsymbol{\mathcal{I}}_{t-1}$, α_u is the update parameter. Simple update without any constraining criteria can corrupt the models and result in tracker drift. To counter this, the model is updated when the model at time t has a high similarity (say $\rho_{ti}^{+} \geq \rho_{th}$)

with the target model. Foreground models are updated when $\rho_{ti}^{+}(\mathbf{m}_{t-1}^{\mathbb{T}+}, \mathbf{m}_{ti}^{\mathbb{C}+}) \geq \rho_{th}$, where ρ_{ti}^{+} represents the Bhattacharyya similarity. The threshold value is kept high to avoid frequent updates to the model thereby preventing tracker drift.

3.3 Localization

During localization object is located in frame $\boldsymbol{\mathcal{I}}_t$ and is defined with $\overset{\star}{\mathbf{bb}}_t^{+}$. A bounding box is parameterized with its state vector and is given as $\mathbf{bb}_t = \mathbf{z}(t) = [x_t, y_t, w_t, h_t, \theta_t]$. Here, (x_t, y_t) are the top left x,y coordinates of the box, w_t, h_t, θ_t are respectively the width, height and orientation of the bounding box. Object localization happens in position, scale and orientation. For this, a search in the neighborhood of $\overset{\star}{\mathbf{bb}}_{t-1}^{+}$ in $\boldsymbol{\mathcal{I}}_t$ is performed. Region proposals and associated candidate models are generated in frame $\boldsymbol{\mathcal{I}}_t, \{t > 0\}$. Search is performed to find the region proposal that maximizes an objective function $\mathcal{J}(.)$. In this work, the objective function used is the similarity $\rho^{\pm}$ as defined in equation 3.11. Object localization now reduces to a region search problem for which many standard algorithms like particle filter [21], PSO [236], DE [236], Firefly [8] exist in literature. Localization is explored with all these approaches. Additionally, a novel approach is presented which inserts new solutions into a firefly population algorithm using RGA [237] operator like crossover. The resulting algorithm is named *FRGA* and is explained in detail in sub-section 3.3.4. Candidate solutions in these algorithm are represented by the 5 dimensional state vector $[x, y, w, h, \theta]$. Set of all the candidate solutions in an iteration τ is referred to as population $\mathbb{P}_\tau$. Objective function associated to i^{th} candidate solution in iteration τ is represented as $\mathcal{J}_i(\tau)$. Their set is given by $\mathbb{J}_\tau = \{\mathcal{J}_i(\tau), i = 1, \ldots, n_{\mathbb{P}}\}$

3.3.1 Probabilistic Approach: Particle Filter

Particle filter or sequential Monte Carlo method estimates the posterior probability density of the object state in a Bayesian formulation. Particles (solutions, $\mathbf{z}$) are initialized randomly in frame $\boldsymbol{\mathcal{I}}_t$.

Let the initial population be $\mathbb{P}_0 = \{\mathbf{z}_i(0), i = 1, \ldots, n_{\mathbb{P}}\}$. Each sample has a weight (objective function) associated to it given by $\mathbb{J}_0 = \{\mathcal{J}_i(0), i = 1, \ldots, n_{\mathbb{P}}\}$. Given the last state $\mathbf{z}(t-1)$ particle filter predicts the next state $\mathbf{z}(t)$. It involves a prediction step and an update step. In the prediction step particles are resampled according to the weight associated to each sample. In the update step the motion model is applied and the measurements are taken to update the weights $\mathcal{J}_t$. Let, the

measurements obtained till $\mathbb{J}_t$ be computed by Bayes rule as follows.

$$p(\mathbf{z}(t)|\mathbb{J}_t) = \frac{p(\mathbb{J}_t|\mathbf{z}(t))p(\mathbf{z}(t)|\mathbb{J}_{t-1})}{p(\mathbb{J}_t)} \tag{3.13}$$

where, $p(\mathbb{J}_t|\mathbf{z}(t))$ can be replaced with $p(\mathcal{J}_t|\mathbf{z}(t))$ as $\mathcal{J}_t$'s are conditionally independent and

$$p(\mathbf{z}(t)|\mathbb{J}_{(t-1)}) = \int_{\mathbf{z}(t-1)} (\mathbf{z}(t)|\mathbf{z}_{(t-1)})p(\mathbf{z}(t-1)|\mathbb{J}_{(t-1)})d\mathbf{z}_{(t-1)} \tag{3.14}$$

where, $p(\mathbf{z}_t|\mathbf{z}_{(t-1)})$ represents the motion dynamics and $p(\mathbb{J}_t|\mathbf{z}(t)$ is the observation density. The state transition is defined by a random walk process given as $p(\mathbf{z}(t)|\mathbf{z}(t-1)) = \mathcal{N}(\mathbf{z}(t); \mathbf{z}(t-1), \Sigma)$. A diagonal covariance matrix is assumed. The Particle with maximum posterior returns the state of the object in frame $\mathcal{I}_t$. Evolutionary localization approaches are explained next.

3.3.2 Evolutionary Approach

This subsection discusses two evolutionary approaches. These are Differential Evolution (DE) and Real-Coded Genetic Algorithm (RGA).

Differential Evolution (DE) is an evolutionary algorithm which finds the optimum solution by iteratively improving candidate solutions. Let $p_{cr} \in [0, 1]$ be the cross-over probability and $w_d \in [0, 2]$ be the differential weight respectively. New candidate solutions are generated using the cross-over operation. For updating the i^{th} solution $\mathbf{z}_i$ three other distinct solution $(\mathbf{z}_{a1}, \mathbf{z}_{a2}, \mathbf{z}_{a3})$ are picked at random from the candidate solution set $\mathbf{z}(\tau)$ at iteration τ. Let $\mathbf{r}_i = [r_i[1], r_i[2] \ldots, r_i[n]]$ be the set of random vector such that $r_i[j] \sim U(0, 1)$. A solution is generated using mutation which is called the donor vector $\mathbf{z}^d$

$$\mathbf{z}_i^d(\tau + 1) = \mathbf{z}_{a1}(\tau) + w_d(\mathbf{z}_{a2}(\tau) - \mathbf{z}_{a3}(\tau)), i =, \ldots, n_{\mathbb{P}} \tag{3.15}$$

A trial vector $\mathbf{z}^{tr}$ is then computed from the donor vector $\mathbf{z}_i^d$ and target vector $\mathbf{z}_i$ using recombination. This is given by

$$\mathbf{z}_i^{tr}[j](\tau + 1) = \begin{cases} \mathbf{z}_i^d[j](\tau + 1) & if\ \mathbf{r}_i[j] \leq p_{cr}\ or\ j = I_{rand} \\ \mathbf{z}_i[j](\tau) & if\ \mathbf{r}_i[j] > p_{cr}\ and\ j \neq I_{rand} \end{cases}$$

where, $I_{rand} \in \{1, \ldots, n\}$. It makes sure that $\mathbf{z}_i^d[j](\tau + 1) \neq \mathbf{z}_i[j](\tau)$. Selection is the last step which is

performed as follows.

$$\mathbf{z}_i(\tau+1) = \begin{cases} \mathbf{z}_i^{tr}(\tau+1) & \text{if } \mathcal{J}(\mathbf{z}_i^{tr}(\tau+1)) < \mathcal{J}(\mathbf{z}_i(\tau)) \\ \mathbf{z}_i(\tau) & \text{otherwise} \end{cases}$$

Genetic Algorithms (GA) are search methods for optimization problems which mimics the principles of natural evolution. Real-coded Genetic Algorithm(RGA) [237] uses the operators of mutation and crossover using real parameters. Let λ_{mu} and λ_{co} be the mutation and crossover probabilities respectively. Simulated Binary Crossover(SBX) operator is used in algorithm. Two children solutions $\mathbf{z}^{c1}(\tau+1), \mathbf{z}^{c2}(\tau+1)$ are generated from two parent solutions $\mathbf{z}_{p1}(\tau), \mathbf{z}_{p2}(\tau)$ as

$$\mathbf{z}^{c1}(\tau+1) = 0.5[(1+\beta)\mathbf{z}_{p1}(\tau) + (1-\beta)\mathbf{z}_{p2}(\tau)] \quad (3.16)$$

$$\mathbf{z}^{c2}(\tau+1) = 0.5[(1-\beta)\mathbf{z}_{p1}(\tau) + (1+\beta)\mathbf{z}_{p2}(\tau)] \quad (3.17)$$

where, β is given as

$$\beta(u) = \begin{cases} 2u^{\left(\frac{1}{\eta+1}\right)} & if\ u < 0.5 \\ \dfrac{1}{2(1-u)^{\left(\frac{1}{\eta+1}\right)}} & \text{otherwise} \end{cases}$$

where, $u \in [0,1]$, η is a non-negative real number called the distribution index. Smaller the value of η the further will be the children from the parents. Let $\mathbb{P}^c(\tau+1) = \{\mathbf{z}_i^{c1}(\tau+1), \mathbf{z}_i^{c2}(\tau+1)\}_{i=1,\ldots,n_{\mathbb{P}}}$ be the children population.

Swarm based algorithms are explained next.

3.3.3 Swarm Approach

Particle Swarm Optimization [85] is a population based algorithm. Here, the solution mimics swarming behaviour according to a few preset rules. Each solution has an associated position($\mathbf{z} \in \mathbb{R}^n$) and velocity($\mathbf{v} \in \mathbb{R}^n$) with it. The initial population is generated by uniformly sampling the search space defined by the lower and upper bounds $\mathbf{bo}_l, \mathbf{bo}_u$ of the solutions respectively. Each particle is moved in the search space by utilizing the personal best position of the particle ($\mathbf{z}_i^{pb}(\tau)$) and the global best($\mathbf{z}^{gb}(\tau)$) of the swarm at iteration τ. At each iteration the particle velocity is updated

which is then used to modify the solution as follows.

$$\mathbf{v}_i[j](\tau+1) = w\mathbf{v}_i[j](\tau) + \phi^c \mathbf{r}_j^1\Big(\mathbf{z}_i^{pb}[j](\tau) - \mathbf{z}_i[j](\tau)\Big) + \phi^s \mathbf{r}_j^2\Big(\mathbf{z}^{gb}[j](\tau) - \mathbf{z}_i[j](\tau)\Big) \quad (3.18)$$

$$\mathbf{z}_i[j](\tau+1) = \mathbf{z}_i[j](\tau) + \mathbf{v}_i[j](\tau+1) \quad (3.19)$$

where, w is the inertia weight which controls the effect of the previous velocity vector on the new vector, $0 \leq \phi^c, \phi^s \leq 4$ are called the cognitive coefficient and social coefficient respectively. $\mathbf{r}_j^1, \mathbf{r}_j^2$ are two random variables variables with uniform distribution ($\in [0,1]$. Value of $\mathbf{z}_i^{pb}(\tau)$ and $\mathbf{z}^{gb}(\tau)$ are updated after this. The algorithm is either run for a fixed number of iterations or till a minimum value of the objective function value is achieved.

Firefly algorithm (FA) [8] is a meta-heuristic approach inspired from the swarming behaviour of tropical fireflies. A brightness value is associated with each firefly. A firefly is drawn to the brighter one. The force of attraction decreases with distance. The brightest firefly always moves in a random direction while the others try to follow the brighter and closer ones. The firefly population $\mathbb{P}_0 = \{\mathbf{z}_i(0), i = 1, \ldots, n_{\mathbb{P}}\}$ is randomly initialized in search space with a population size $n_{\mathbb{P}}$. Brightness (objective function) values of fireflies is computed as $\mathbb{J}_0 = \{\mathcal{J}_i(0), i = 1, \ldots, n_{\mathbb{P}}\}$. Consider the population $\mathbb{P}_\tau$ in the $(\tau)^{th}$ iteration with brightness values $\mathbb{J}_\tau = \{\mathcal{J}_i(\tau), i = 1, \ldots, n_{\mathbb{P}}\}$. The attraction $\beta_{pq}(\tau)$ between fireflies $\mathbf{z}_p(\tau)$ and $\mathbf{z}_q(\tau)$ is defined as

$$\beta_{pq}(\tau) = \frac{\beta_0}{1 + \eta\|\mathbf{z}_p[j](\tau) - \mathbf{z}_q[j](\tau)\|_2^2} \quad (3.20)$$

where, β_0 is the attraction between coincident fireflies, η is the light absorption coefficient

If $\mathcal{J}_q(\tau) > \mathcal{J}_p(\tau)$, then the velocity $\boldsymbol{\upsilon}_{pq}^z(\tau)$ of $\mathbf{z}_p(\tau)$ induced by $\mathbf{z}_q(\tau)$ is given by

$$\boldsymbol{\upsilon}_{pq}^z(\tau) = \beta_{pq}(\tau)(\mathbf{z}_q(\tau) - \mathbf{z}_p(\tau)) + \alpha_j(a_j - 0.5) \quad (3.21)$$

where, α_j is a randomization parameter and $a_j \in [0,1]$. The first term in equation 3.21 induces motion towards the brighter firefly by scaling it with attraction. The second term introduces a random but small distraction from this direction of motion. Let, $\mathbf{J}(\tau)$ be the set of all firefly indices such that $\mathcal{J}_q(\tau) > \mathcal{J}_p(\tau)\ \forall\ p \in \mathbf{J}(\tau)$. The next position $\mathbf{z}_p(\tau+1)$ of the p^{th} firefly is derived from the net motion $\boldsymbol{\upsilon}_p^z(\tau)$ induced on it and is given as

$$\boldsymbol{\upsilon}_p^z(\tau) = \sum_{q \in \mathbf{J}(\tau)} \boldsymbol{\upsilon}_{pq}^z(\tau) \quad (3.22)$$

$$\mathbf{z}_p^{new}(\tau+1) = \mathbf{z}_p(\tau) + \boldsymbol{\upsilon}_p^z(\tau) \quad (3.23)$$

Diversification can be brought with Real-Coded Genetic Algorithm (RGA) operations as explained next.

3.3.4 Firefly RGA (FRGA)

The firefly algorithm (FA) searches for optimal solutions in a population based approach. Researchers have proposed hybrid firefly algorithms by introducing genetic operators in creating new fireflies [86]. These works have mostly used differential evolution [236] where one part of the population changes position using evolutionary techniques (GA or RGA [237]) and the other updates positions using the basic FA. However, these methods do not consider the velocity vectors in breeding fireflies and genetic operations are performed directly on the solution vectors. Settles *et al.* [242] have introduced the contribution of velocities in crossover operation in context of hybrid PSO. Introduction of particle velocities in crossover has shown improvements over differential evolution schemes [236]. Accordingly, this work proposes a hybrid firefly algorithm that breeds fireflies with parent velocity information. The proposed approach is described as follows.

Firefly population $\mathbb{P}_0 = \{\mathbf{z}_i, i = 1, \ldots, n_{\mathbb{P}}\}$ is randomly initialized in search space. Brightness (objective function) values of fireflies are computed as $\mathbb{J}_0 = \{\mathcal{J}_i(0), i = 1, \ldots, n_{\mathbb{P}}\}$. Each firefly represents a solution bounding box parameterized by $\mathbf{z}$. A firefly always gets attracted to a brighter one. Attraction decreases exponentially with distance and increases with brightness (equation 3.20). Motion is induced in the fireflies due to this attraction. It results in newer (and possibly improved) solutions. Incremental improvement is achieved in each iteration τ. One such iteration is elaborated next.

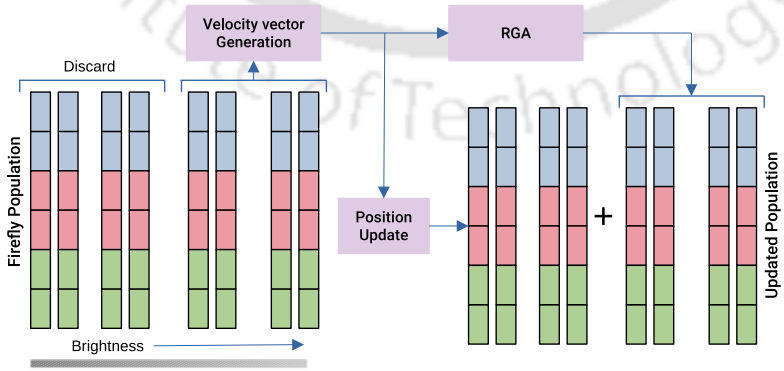

Figure 3.5: Proposed hybrid firefly algorithm for tracking. Object bounding box parameters are combined in solution vector containing centroid (red), scaling in width and height (green) and orientation (blue). A part of population is updated following basic firefly algorithm. The remaining population is bred using velocity propelled crossover operator [250].

.

As illustrated in Figure 3.6, fireflies in the population $\mathbb{P}_\tau$ are first sorted in ascending order of brightness values. The worst $n_d = \lfloor br \times n_\mathbb{P} \rfloor$ ($br \in (0,1)$ is the breeding ratio) fireflies are discarded to form the elite population $\mathbb{P}^e_\tau = \{\mathbf{z}^e_j(\tau); j = 1 \ldots, n_e\}$ ($ne = n_\mathbb{P} - n_d$). The net motion vectors $\boldsymbol{\upsilon}^{ze}_i(\tau)$ ($i = 1, \ldots, n_e$) of the elite fireflies (in $\mathbb{P}^e_\tau$) are estimated using equation 3.22. The new positions of elite fireflies are updated using their motion vectors (equation 3.23) to form the population .

$$\mathbb{P}^e(\tau) = \{\mathbf{z}^e_i(\tau); i = 1, \ldots, n_e\} \tag{3.24}$$

The population gap generated by discarding n_d fireflies in τ^{th} iteration is filled up by breeding fireflies in τ^{th} iteration. Parent fireflies are selected for offspring generation. Let, $\mathbf{z}^e_{p1}(\tau)$, $\mathbf{z}^e_{p2}(\tau) \in \mathbb{P}^e(\tau)$ be two parents chosen by the process of tournament selection [251]. The offsprings $\mathbf{z}^{ebr}_{p1}(\tau)$ and $\mathbf{z}^{ebr}_{p2}(\tau)$ are generated through crossover operation combined with corresponding parent velocity information. This is performed using the following equations.

$$\mathbf{z}^{ebr}_{p1}[j](\tau) = \frac{1}{2}\Big(\mathbf{z}^e_{p1}[j](\tau) + \mathbf{z}^e_{p2}[j](\tau)) - \phi_p \boldsymbol{\upsilon}^{ze}_{p1}[j](\tau)\Big) \tag{3.25}$$

$$\mathbf{z}^{ebr}_{p2}[j](\tau) = \frac{1}{2}\Big(\mathbf{z}^e_{p2}[j](\tau) + \mathbf{z}^e_{p1}[j](\tau)) - \phi_{p2} \boldsymbol{\upsilon}^{ze}_{p2}[j](\tau)\Big) \tag{3.26}$$

where, ϕ_{p1} and ϕ_{p2} are random variables drawn from the uniform distribution $\in [0,1]$. The parent fireflies are repeatedly sampled using tournament selection till n_d firefly offspring are generated. These offsprings form the population $\mathbb{P}^{ebr}(\tau)$. The brightness values of $\mathbb{P}^e(\tau)$, $\mathbb{P}^{ebr}(\tau)$ are computed and compared with that of $\mathbb{P}_\tau$ to choose the brightest $n_\mathbb{P}$ fireflies. These fireflies (best solutions) form the population $\mathbb{P}_{\tau+1}$ (with set of brightness values $\mathbb{J}_{\tau+1}$) for the next iteration.

For estimating $\overset{\star}{\mathbf{bb}}^{+}_t$ from the solution vectors (fireflies) in the search space, brightness value of firefly is defined as given in equation 3.11. Best solution at the end of n_τ iterations is identified as the object state $\overset{\star}{\mathbf{bb}}^{+}$ in the given frame $\boldsymbol{\mathcal{I}}_t$.

Similarly, object can be localized with the help of motion information computed with optical flow. It is already established in literature that appearance and motion are complementary components [77] and their combination in an ensemble framework can enhance overall performance [77]. Additional processing in the form of motion correction is used along with *FRGA* to enhance performance and is explained next.

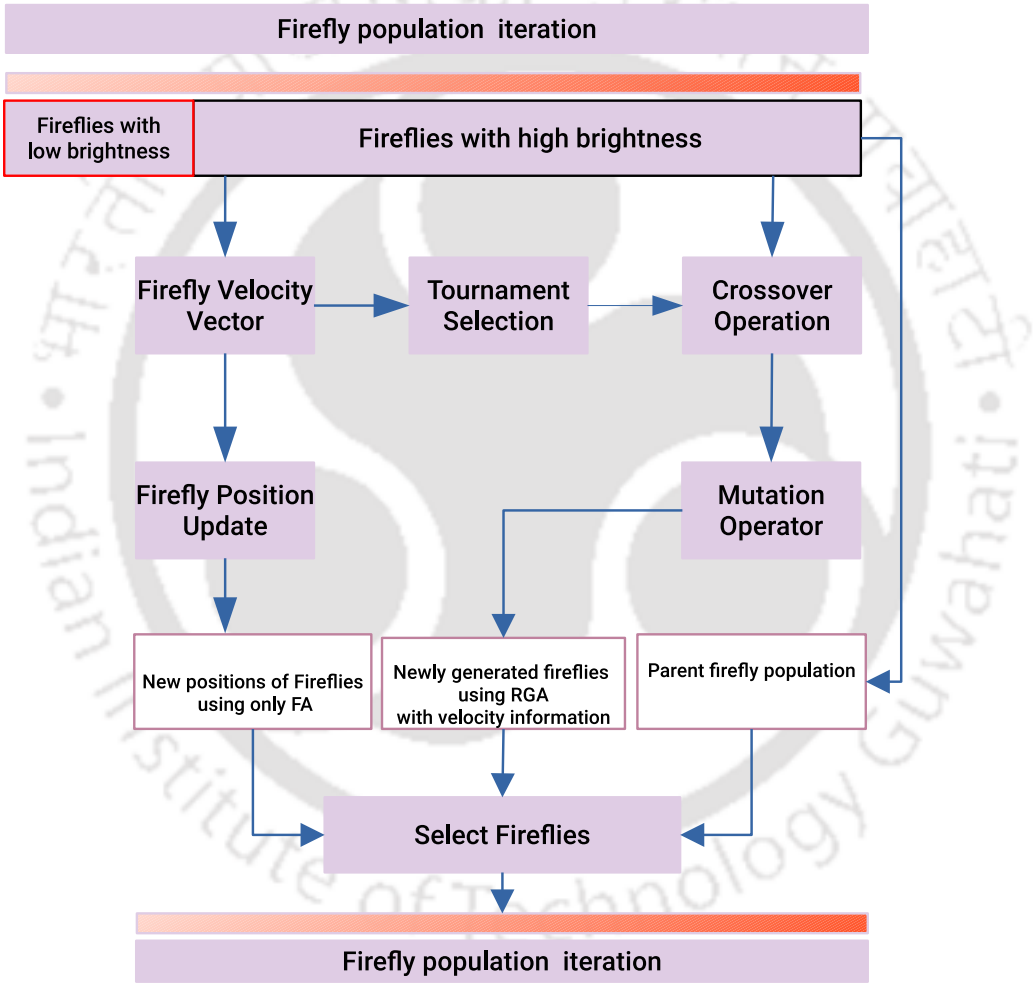

Figure 3.6: Illustrating the *FRGA* approach of breeding fireflies.

3.4 Motion Correction

Estimating the motion of object pixels in a frame $\mathcal{I}_{t-1}$ can help in locating object in the next frame $\mathcal{I}_t$. Inter-frame motion is computed with optical flow algorithms. Popular methods include Farneback [91] and Horn–Schunck [91]. Lucas Kanade tracker is a classical example of a tracker built on optical flow based approach. For simplicity classical method is employed for flow estimation in this modification. Appearance feature might be similar across the target and the distractors. However, the motion information associated with them might be different. A re-ranking of solutions from the appearance component is performed with motion information to improve the prediction.

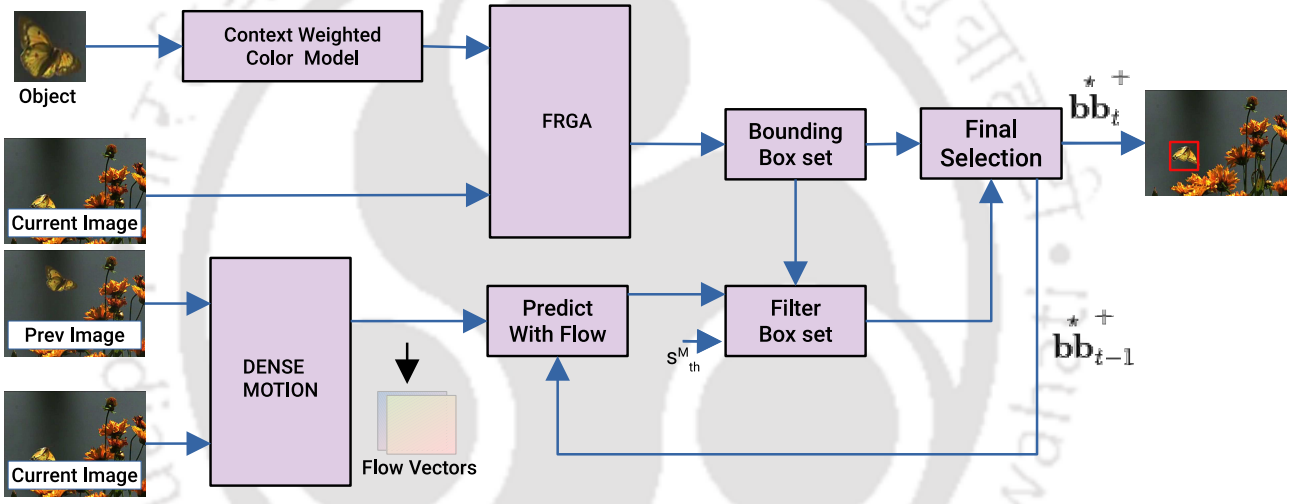

Figure 3.7: Tracking of object in frame $\mathcal{I}_t$ achieved with both appearance and motion components

Per-pixel dense motion is computed from two consecutive image pairs $\mathcal{I}_{t-1}$ and $\mathcal{I}_t$. Let $\overset{\star}{\mathbf{bb}}^{+}_{t-1}$ be the predicted bounding box from the last frame $\mathcal{I}_{t-1}$. Let the dense motion vector matrices from the computed be $\mathbf{U}_{t-1}, \mathbf{V}_{t-1}$. Here $\mathbf{U}_{t-1}, \mathbf{V}_{t-1}$ represent the set of motion vectors in horizontal and vertical directions respectively.

Let the set of all pixel coordinates belonging to bounding box $\overset{\star}{\mathbf{bb}}^{+}_{t-1}$ be defined as $\overset{\star}{\mathcal{P}}^{+}_{t-1}$. Next frame position of every pixel in $\overset{\star}{\mathcal{P}}^{+}_{t-1}$ can be predicted from motion as follows

$$(x,y)^M_t = (x,y)_{t-1} + (\mathbf{U}_{t-1}[x,y], \mathbf{V}_{t-1}[x,y]);\ (x,y)_{(t-1)} \in \overset{\star}{\mathcal{P}}^{+}_{t-1} \tag{3.27}$$

where, $\mathbf{U}_{t-1}[x,y], \mathbf{V}_{t-1}[x,y]$ are the motion vector components at position (x,y). Let $\mathcal{P}^M_t$ represent pixel position set $\mathcal{P}^M_t = \{(x,y)^M_t\}$. Each candidate solution from appearance component represent a bounding box with a score assigned to it. Score is computed from the objective function defined in

equation 3.11. Let the score of all the candidate solutions be given as

$$\mathbb{S}_t = \{s_{ti} = \mathcal{J}_{ti} = \rho_{ti}^{\pm}, i = 1, \ldots, n_{\mathbb{P}}\} \tag{3.28}$$

Here, s_{ti} is the score of the i^{th} candidate bounding box. Predictions are now available from both appearance and motion components. Final decision is made on the degree of agreement (s_{th}^M) between the two components. It is quantified as the amount of overlap between the candidate solution from appearance component and the predicted solution from motion component as follows.

$$\rho_{ti}^M = \frac{|\overset{\star}{\mathcal{P}}{}_{t-1}^{+} \cap \mathcal{P}_{ti}^{+}|}{|\overset{\star}{\mathcal{P}}{}_{t-1}^{+} \cup \mathcal{P}_{ti}^{+}|} \tag{3.29}$$

where, $\overset{\star}{\mathcal{P}}{}_{t-1}^{+}$ is the set of pixel locations predicted by the motion component and $\mathcal{P}_{ti}$ is the set of pixels of the i^{th} candidate box from the appearance component. New scores ($\mathbb{S}_t^M$) are computed for all candidate solutions of appearance component using equation 3.29. Best solutions are identified as the ones that satisfy a minimum threshold criteria s_{th}^M as follows

$$\mathbb{S}_t^M = \{s_{ti}^M = \rho_{ti}^M, \rho_{ti}^M > s_{th}^M\} \tag{3.30}$$

Here, $\mathbb{S}_t^M$ is the set obtained after performing re-ranking with motion information. If $\mathbb{S}_t^M$ is empty then the algorithm reverts back to appearance component. Let $\mathbf{J}_t = \{i \parallel \rho_{ti}^M \geq s_{th}^M; i = 1, \ldots, n_{\mathbb{B}}\}$ is the set of indices that define bounding boxes $\mathbb{B}_t^M = \{\mathbf{bb}_{ti}, i \in J_t\}$ with a minimum agreement s_{th}^M. The optimum candidate box $\overset{\star}{\mathbf{bb}}{}_t^{+} = \mathbf{bb}_{ti\star}^{+}$. Here $i^\star$ is selected as follows.

$$i\star = \begin{cases} \underset{i \in [1,\ldots,n_{\mathbb{P}}]}{\operatorname{argmax}}(s_{ti}), s_{ti} \in \mathbb{S}_t, & \text{if } |\mathbb{S}_t^M| = 0 \\ \underset{i \in \mathbf{J}_t}{\operatorname{argmax}}(s_{ti}^M), s_{ti}^M \in \mathbb{S}_t^M, & \text{otherwise} \end{cases}$$

Here, $s_{th}^M \in [0, 1]$ controls the influence of motion component in the ensemble. A value of 1 implies only appearance component is used. The proposed approach has been tested on standard datasets. Block diagram of the combined approach is given in Figure 3.7 and the pseudocode for the same is provided in algorithm 1.

Algorithm 1 $FRGA + MC$

Input $\boldsymbol{\mathcal{I}}_0$, $\mathbf{bb}_0^+$

Output $\overset{\star}{\mathbf{bb}}{}_t^+$

Define $\boldsymbol{\mathcal{I}}^Q = extractROI(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_0^+)$

$\mathbf{m}_0^{\mathbb{T}+}, \mathbf{m}_0^{\mathbb{T}-}$ = histModel($\boldsymbol{\mathcal{I}}_0$, $\mathbf{bb}_0^+$) ▷ refer equation 3.5

while $t \leq (n^T)$ **do** ▷ Loop through sequence

  $(\mathbf{U}_{(t-1)}, \mathbf{V}_{(t-1)}) = \phi_{DenseOpticalFlow}(\boldsymbol{\mathcal{I}}_{t-1}, \boldsymbol{\mathcal{I}}_t)$

  $\overset{\star}{\mathbf{bb}}{}_{(t-1)}^+ = [x_{(t-1)}, y_{(t-1)}, w_{(t-1)}, h_{(t-1)}, \theta_{(t-1)}]$

  $\boldsymbol{\mathcal{I}}_t^S = extractROI(\boldsymbol{\mathcal{I}}_t, \overset{\star}{\mathbf{bb}}{}_{t-1}^+)$

  $\overset{\star}{\mathcal{P}}{}_t^+ = \{(x,y)_{t-1} + (\mathbf{U}_{t-1}[x,y], \mathbf{V}_{t-1}[x,y]), (x,y)_{t-1} \in \overset{\star}{\mathcal{P}}{}_{(t-1)}^+\}$

  $\mathbb{P}_0$ = initalPopulation($\boldsymbol{\mathcal{I}}_t$, $\overset{\star}{\mathbf{bb}}{}_{t-1}^+$)

  while $\tau \leq n_\tau$ **do** ▷ iteration

    $\mathbb{P}_{t\tau}^C$ = childrenPopulation($\mathbb{P}_{(\tau)}$)

    while $i =\leq n_{\mathbb{P}}$ **do** ▷ Iterating through population

      $\mathbf{m}_{ti\tau}^{\mathbb{C}+}, \mathbf{m}_{ti\tau}^{\mathbb{C}-}$ = histModel($\boldsymbol{\mathcal{I}}_t$, $\mathbb{P}_{ti\tau}^C$) ▷ (refer eq 3.7,3.8)

      $s_{ti(\tau)} = \rho_{ti}^+(\mathbf{m}_0^{\mathbb{T}+}, \mathbf{m}_{ti(\tau)}^{\mathbb{C}+}) \times (1 - \rho_{ti}^+(\mathbf{m}_0^{\mathbb{T}+}, \mathbf{m}_{ti(\tau)}^{\mathbb{C}-}))$

    end while

    $\mathbb{P}_{t\tau}$ = newPopulation($\mathbb{P}_{t(\tau)}, \mathbb{P}_{t(\tau)}^C, \mathbb{S}_{t(\tau)}$)

  end while

  $\mathbb{S}_{tn_\tau}^M = \frac{|\mathcal{P}_t^M \cap \mathcal{P}_{tn_\tau}|}{|\mathcal{P}_t^M \cup \mathcal{P}_{tn_\tau}|}$

  Generate the set $\mathbf{J}_t$

  if $\mathbf{J}_t \neq \emptyset$ **then**

    $\overset{\star}{\mathbf{bb}}{}_t^+ := \mathbf{bb}_{ti^\star}^+ \; ; i^\star = \underset{j \in \mathbf{J}_t}{\operatorname{argmax}} \; s_{ti}^M \in \mathbb{S}_t^M$

  else

    $\overset{\star}{\mathbf{bb}}{}_t^+ := \mathbf{bb}_{ti^\star}^+ \; ; i^\star = \underset{j \in [1, n_{\mathbb{P}}]}{\operatorname{argmax}} \; s_{ti} \in \mathbb{S}_t$

  end if

end while

3.5 Experiments & Results

Evaluation is performed on $VOT2018$ dataset with VOT toolkit and on $OTB2015$, $UAV123$ datasets with GOT-10K toolkit. For VOT, Accuracy, Robustness and Expected Average Overlap are the measures reported. Accuracy and EAO are better with higher value (↑) and robustness is better when the value is lower (↓). For $OTB2015$ and $UAV123$ success and precision plots are reported for comparison. More on the datasets, performance measures and benchmarking tools are covered in sections 2.4 and2.5.

3.5.1 Quantitative Analysis

Experiments were conducted to compare and contrast the two different models and the localization algorithms. Two tracking algorithms were proposed with particle filter based localization component and color intensity values for modeling. In one approach only the foreground bounding box $\mathbf{bb}_0^+$ is used in object modeling and in the second approach contextual information from $\mathbf{bb}_0^-$ is utilized for the same. Experimental results shown in table 3.1 shows that the model with contextual information outperformed the one with only a foreground component. Improvement was seen across all the performance measures indicating the superiority of the context weighted model.

In the second experiment context weighted model is used across all the trackers, but different localization algorithms are used. Localization algorithms tested include particle filter, particle swarm optimization, real coded genetic algorithm, differential evolution and firefly algorithm. Experimental results in Table 3.2 shows that the firefly algorithm provides the best accuracy among the different approaches making it a suitable candidate to perform further modifications. Modifications to the firefly algorithm was performed with breeding operation. Performance of tracking algorithm varied with breeding ratio br as shown in Table 3.4. The best performance was observed with a value of $br = 0.2$. At this value, the tracker gave the best accuracy and expected overlap while coming second in robustness. Ensembling was performed on the above approach with motion information from optical flow. Experimental results in table 3.5 show that the tracker performing similar to the appearance only one. Accuracy and expected average overlap values were degraded Accuracy (from $from\ 0.40\ to\ 0.397$ and EAO ($from\ 0.88\ to\ 0.86$) slightly and small improvement was observed on robustness ($from\ 5.521\ to\ 5.245$).

.

Ensemble approach with appearance and motion is also evaluated on $VOT2018$ dataset and is

Name	A ↑	R ↓	EAO ↑
$CD + PF$	0.244	3.947	0.065
$CWCD + PF$	0.344	2.831	0.106

Table 3.1: Performance comparison of particle filter based tracking with color histogram (CH) and context weighted color histogram (CWCH). Context weighting significantly improves tracking performance.

Name	A ↑	R ↓	EAO ↑
$CWCD + PF$	0.344	2.831	0.106
$CWCD + PSO$	0.396	8.493	0.068
$CWCD + RGA$	0.396	6.236	0.081
$CWCD + DE$	0.392	6.239	0.081
$CWCD + FA$	0.407	5.304	0.081

Table 3.2: Performance comparison of different localization techniques on $VOT2018$ dataset. Firefly based approach performed the best among the lot.

Name	A ↑	R ↓	EAO ↑
$CWCD + FA$	0.407	5.304	0.081
$CWCD + FRGA$	0.401	5.521	0.088

Table 3.3: Performance comparison of different localization techniques with Firefly and FRGA on $VOT2018$ dataset.

Breeding Ratio	A ↑	R ↓	EAO ↑
0.1	0.391	5.448	0.085
0.2	0.401	5.521	0.088
0.4	0.373	7.350	0.071
0.6	0.380	7.977	0.070
0.8	0.376	8.395	0.066

Table 3.4: Performance comparison of *FRGA* approach with different breeding ratios on $VOT2018$ dataset with Accuracy (A), Robustness (R) and Expected Average Overlap (EAO). Evaluation is done on 60 sequences available in the dataset.

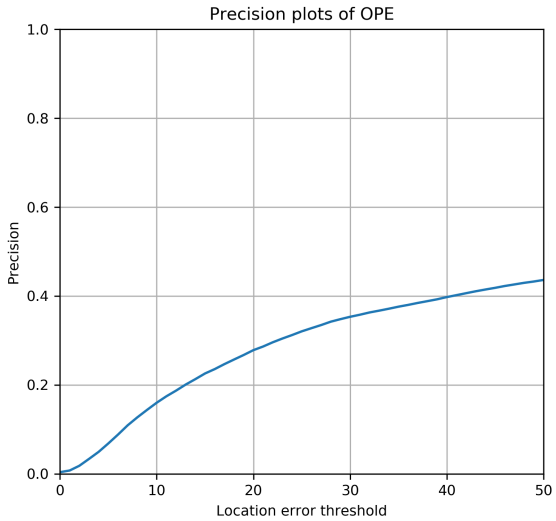

Figure 3.8: Precision plot of proposed tracker on $OTB2015$ dataset.

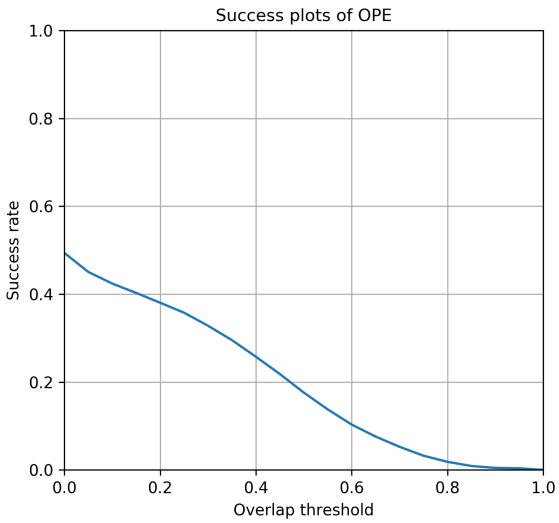

Figure 3.9: Success plot of proposed tracker on $OTB2015$ dataset.

compared with the appearance only approach as given in table 3.5. Proposed ensemble was able to improve robustness with negligible reduction in accuracy and EAO.

Mode	A ↑	R ↓	EAO ↑
$CWCD + FA$	0.407	5.304	0.081
$CWCD + FRGA$	0.401	5.521	0.088
$CWCD + FRGA + MOTION$	0.397	5.245	0.086

Table 3.5: Performance comparison of $CWCD + FRA$ with and without the motion component.

The tracker is benchmarked on challenging sequences from the $VOT2018$ dataset. The tracker performance is compared against baseline algorithms like MatFlow [252], IVT [253], MIL [36], BST [254], DSST [62], FoT [62] and BDF [252] available in VOT toolkit. The comparative performance analysis is presented in Table 3.5.

Table 3.6 shows that our proposed trackers performance is in line with other classical trackers. Apart form $VOT2018$, ensemble approach is tested on $OTB2015$ and $UAV123$ datasets. Evaluation of the proposed $CWCD + FRGA + MOTION$ algorithm on $OTB2015$ dataset with precision plot and success plot is shown in figures 3.8 and 3.9 respectively. For comparison with other trackers on $OTB2015$ dataset, the plots are shown in Appendix A.2.

Evaluation on $UAV123$ dataset is shown in figures 3.10 and 3.11. Performance is negatively impacted as the evaluations strategy follows no re-initialization rule and the current version of proposed

Name	**Accuracy↑**	**Robustness↓**	**EAO↑**
Matflow	0.386	4.973	0.091
BDF	0.355	4.476	0.092
BST	0.282	4.086	0.118
DSST	0.388	5.43	0.079
FoT	0.388	4.19	0.129
CWCD+FA	0.407	5.304	0.081
CWCD+FRGA	0.401	5.521	0.088
CWCD+FRGA+MOTION	***0.397***	***5.245***	***0.086***
IVT	0.382	6.093	0.076
MIL	0.383	3.933	0.117

Table 3.6: Performance comparison on $VOT2018$ dataset with the baseline scheme in $VOT2018$ toolkit. Evaluation is done on 60 sequences available in the dataset. Proposed tracker is highlighted in bold and italics font.

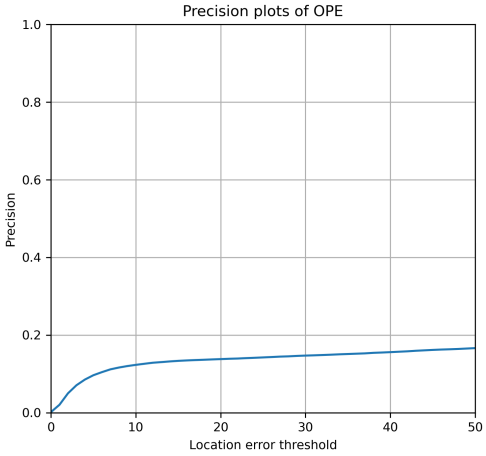

Figure 3.10: Precision plot of proposed tracker on $UAV123$ dataset.

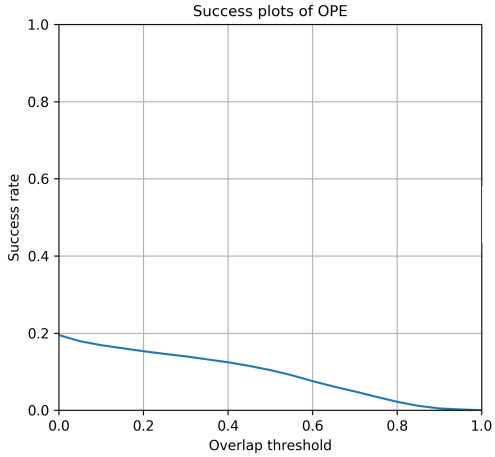

Figure 3.11: Success plot of proposed tracker on $UAV123$ dataset.

tracker doesn't have track re-acquisition module once the track is lost.

Parameters chosen for implementation – For differential evolution, mutation and crossover probability was taken as 0.9. For firefly algorithm $\beta_0 = 2.0$ and $\eta = 0.01$. For iteration based algorithms an iteration number of 10 with 25 particle populations where considered. The search space was defined as $[w_{min}, w_{max}] = [0.7 \times \overset{\star}{w}_{t-1}, 1.3 \times \overset{\star}{w}_{t-1}]$ for scaling in width and $[h_{min}, h_{max}] = [0.7 \times \overset{\star}{h}_{t-1}, 1.3 \times \overset{\star}{h}_{t-1}]$ for scaling in height. For translation, search space was defined within 30 pixel variation and for orientation $[\theta_{min}, \theta_{max}] = [0, \frac{\pi}{2}]$.

3.5.2 Qualitative Analysis

Object model with contextual information showed better overall tracking performance compared to the foreground only model. Visual results of object tracking on some of these sequences are presented in figure 3.12. Ensemble was able to adapt to geometric transformations like scale change and rotation effectively as the model used is invariant to scaling and rotation.

3.6 Discussions

Image color intensity distribution has been widely used in tracking. This feature is easier to compute and is somewhat robust to scaling and in-plane rotations. This chapter explored the color distribution feature to construct the object appearance model. It was observed that rectangular bounding box based annotation of non-rectangular objects and background clutter may lead to tracker drift (or track loss). This can be attributed to the presence of immediate background information in the learned object color distribution. To this end, this work proposes a context weighted color distribution that promotes the object colors while suppressing the ones coming from immediate backgroud.

A novel objective function for target localization is also proposed while considering the information of both object and background. This objective function is maximized for object localization using probabilistic and meta-heuristic search strategies. Object localization was attempted using particle filter, evolutionary search strategies (DE and RGA) and swarming techniques (PSO and FA). It was observed that the Firefly Algorithm provided the best search performance. In Firefly Algorithm, update of a solution is controlled by all solutions that are better than it. In comparison, for a PSO algorithm there is only one global best and local best in each iteration. Each solution tends to move in a directin which is collectively decided by the local and global best. Since updates are controlled by multiple solutions rather than a single one, this improves exploration and results in better localization. Firefly algorithm is found to be suitable for multi-modal problems as mentioned in [86]. Motivated by the success of breeding swarm strategies, a velocity propelled real-coded genetic operator based modification to FA was proposed. The predictions of this proposed breeding swarm algorithm (FRGA) was further re-ranked based on inter-frame motion information. The proposed approach was observed to provide better or competitive performance against seven classical baseline tracking algorithms on three standard datasets.

Color distribution is a global feature and is often considered to be invariant to local variations

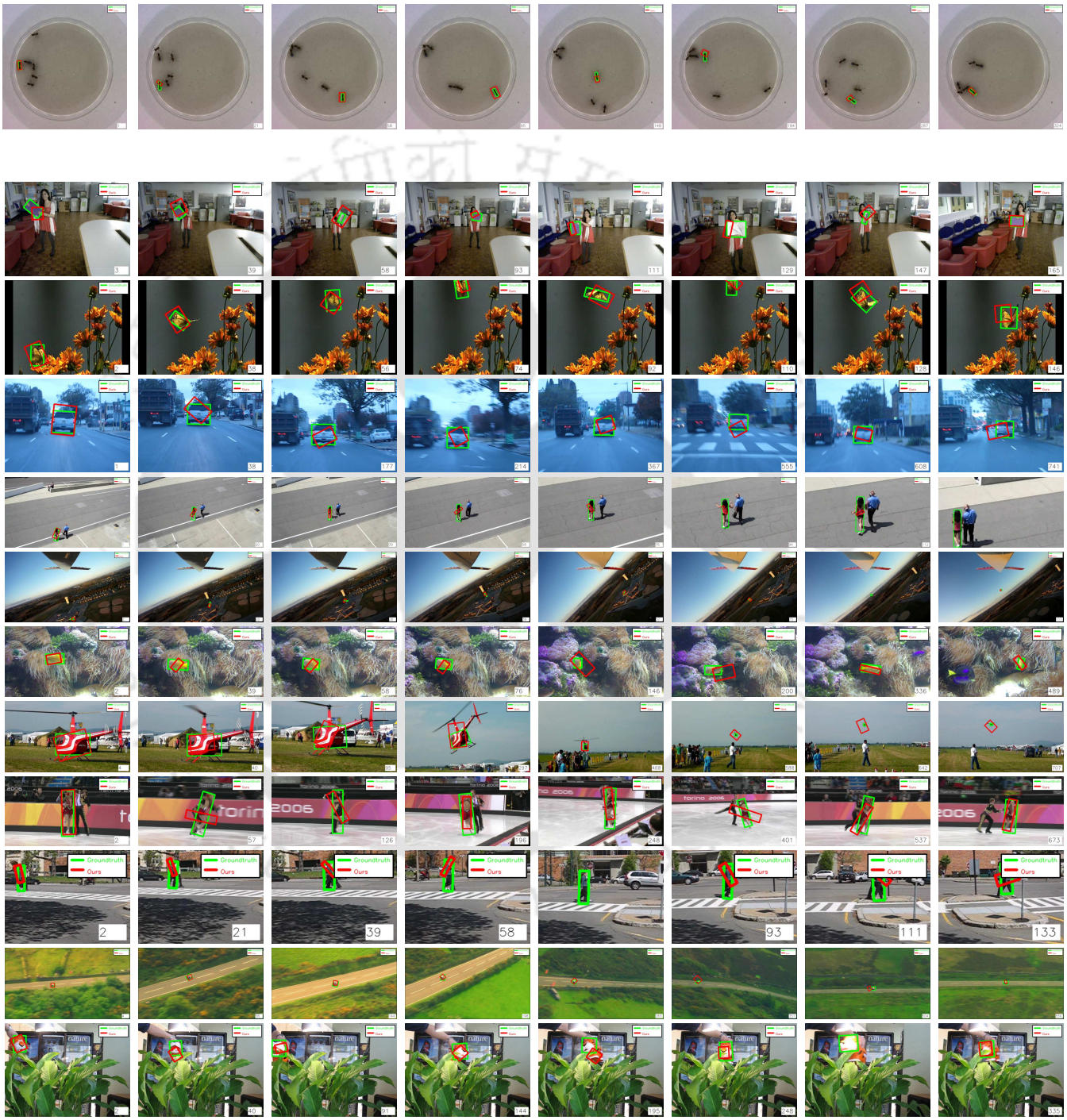

Figure 3.12: Each row shows outputs of visual tracking on sequences from $VOT2018$ dataset. Groundtruth is marked in green and tracker prediction is colored red.

within the object. For example, two objects with different structures can have the same color distribution. Local structural information plays an important role in describing an object. Such structure information based descriptors may bring further differentiation between objects having same color distribution. For example, local information of object appearance structure can be extracted from object patches. Accordingly, sparse representation based object appearance encoding is explored. The sparse representation based object model is presented next in Chapter 4.

4

Sparse Representation based Tracking

Publications

- **Mathew Francis**, Prithwijit Guha, "Object Tracking with Classification Score Weighted Histogram of Sparse Codes", *International Conference on Pattern Recognition and Machine Intelligence, 2017*.

Contents

Overview

Color distribution computed from the entire target object region is a global feature. It does not capture local variations within the object region. Accordingly, this chapter focuses on developing sparse representation based object models for capturing local structural information of the target. Sparse code coefficients of patch vectors are generated with a learned dictionary and their distribution is used to model the object. For tracking, it is important to discriminate the object from its background. This discrimination is brought with the addition of explicit background dictionary. Two models are proposed in this chapter. The first model uses a distribution of sparse codes in a generative framework. The second one has an additional binary classifier component in a hybrid framework to improve discrimination. The learned binary classifier provides weights to the sparse codes of patches extracted from pixel positions according to their belongingness to the object. It is similar to weighing the pixel intensity values in a color distribution based model. Based on the model framework used, novel objective functions are defined to localize object with FRGA algorithm during the tracking phase. In addition to this, motion cues from optical flow computation is used to refine the localization process. The proposed trackers are evaluated on standard datasets like $VOT2018$, $OTB2015$ *and* $UAV123$.

4.1 Sparse Representation based Trackers: An Introduction

This chapter explores the sparse representation of signal to define an object model for tracking. In sparse coding, given an over-complete dictionary any signal can be represented as a linear combination of only few dictionary elements (or atoms). Such a representation of the vectorized object patches can capture structural information of the target object being tracked. Also, it is reported to be robust to occlusion and noise present in a tracking scenes [255]. A global model like color distribution (learned from entire object region) is easy to compute but loses important structural cues of the object. Such local information is helpful in long-term tracking. Object image patches are capable of storing the structural information. This is then encoded with sparse coding operation with a learned dictionary. A distribution of sparse code (DSC) coefficients is computed to model the object. A similar sparse code based model named histogram of sparse codes (HSC) [256] is successfully used in object detection literature. Two object model formulations are explored in this work. First, a generative model developed purely on sparse representation of object patch vectors. Second, a combination of sparse codes and a binary classifier in a hybrid framework. Appearance modeling based on sparse cod-

ing (AMSC) and Target Search based on Sparse Representation (TSSR) is widely explored in tracking as evident from the literature [257].

Mei *et al.* [23] were the first to formulate tracking in a sparse representation framework. Here, tracking was formulated as a least squares minimization problem with l_1 regularization. A basis set consisting of object templates and trivial templates were respectively used to model object appearance. Sparse representation based object models were particularly famous for their robustness against occlusion [16]. All sparse representation models were built on a dictionary which were either defined with object template or learned from it. Some of the popular dictionary learning techniques include K-SVD [258], its discriminative variants [259], proximal alternating method [260], k-selection [46] among others. Dictionary can be defined with object template [23] or on object patches [261]. Many algorithms propose novel dictionary learning procedures [261], object modeling techniques [256, 262], and novel objective functions [257]. Sparse coding techniques use the dictionary of atoms for representing the object. Some of the popular ones include Orthogonal Matching Pursuit (OMP) [263], Focal Undetermined System Solver (FOCUSS) [263] among others. Mei *et al.* [23] used sparse representation of object template and re-projection error for tracking. Jia *et al.* [116] introduced the alignment pooling for feature computation from sparse codes. The reconstruction error was used as the measure of the dissimilarity between object and candidate models (constructed from candidate region proposals) to identify the optimum candidate.

Most of the algorithms used object specific information in tracking [257]. However, there are approaches that used background information as well. Liu *et al.* [46] highlighted the importance of discrimination power of a dictionary in tracking vis-a-vis its reconstruction power. Shengping *et al.* [257] highlighted that discriminative dictionary is important to achieve high tracking accuracy. Discrimnative dictionary learning itself is a widely researched topic [259, 264]. The dictionary can be learned form object as well as background during modeling. Ning *et al.* [40] has shown that the use of contextual information improves tracking performance. Bai *et al.* [48] used background information to eliminate outlier candidates based on their sparse codes. Zhang *et al.* [47] used discriminative information computed from foreground and background atoms for tracking. Classifiers can also be used to improve discrimination [265]. Wright *et al.* [265] successfully learned classifier on sparse representation for the recognition task. Sparse representation provides robustness to occlusion in tracking [255]. Wang *et al.* [266] used PCA subspace learning with sparse codes to handle occlusion.

It can also be addressed with part-based models as given in literature [102, 267]. Adam *et al.* [102] handled occlusion with a part based approach. A target template is divided into parts and each part is tracked independently and their predictions are combined for locating the object. Yang *et al.* [267] used a bag of features approach in handling occlusions. The object localization stage uses the learned dictionary and the models with Target Search based on Sparse Representation (TSSR) and other approaches in tracking [257].

The TSSR methods include ℓ_1 and ℓ_2 minimization as used in trackers [23, 253, 268] during the localization stage. The candidate models are computed from the generated region proposals. Objective function is defined on object and candidate models for the localization algorithm. Mei *et al.* [23] performed localization with ℓ_1 minimization in a particle filter framework. The particle filter framework is also known as the sequential Monte Carlo [269] method for state estimation and is widely explored in tracking [23]. Zhang *et al.* [270] exploited joint sparsity among particles in sparse coding. Zhang *et al.* [45] used structural information by exploiting the overlap among the particles for tracking. Often the objective functions are not differentiable while making it impossible to use gradient based approaches [4]. Probabilistic [23] and meta-heuristic approaches [24] are used in such cases.

In this work, two object models are explored. One is defined in a generative framework and other in the hybrid framework. Image patches hold structural information of the object and is used in learning a dictionary to model the object. A single object template might be a rigid model to accommodate appearance changes as compared to a smaller patch. To improve discrimination, contextual information is used. A dictionary is learned explicitly to represent the background. Overall dictionary has atoms representing both object and background. Sparse codes are computed with OMP and their distribution is used as the model. In the generative framework, explicit target model is not used and the candidate models are computed during localization. Novel objective function that uses the non-overlap of foreground and background components of the candidate is defined for localization. In the hybrid framework, a part based approach is followed which divides the object into predefined non-overlapping parts. Object models are computed for each part and a collection of them defines the overall model. To improve discrimination of the model, binary SVM classifier trained on the sparse code vectors is used to weight the sparse coefficients in the model. Objective function based on Bhatacharyya similarity is defined between the target and candidate models. For localization, a

meta-heuristic approach *FRGA* introduced in section 3.4 (Chapter3) is used with the proposed objective functions. Motion correction based reranking is performed with values computed with optical flow information as discussed in section 3.4 (Chapter3). An ensemble of motion and appearance components form the tracking algorithm. Performance evaluation is reported on $VOT2018$, $OTB2015$ and $UAV123$ datasets. The main contributions of this work include the following.

- A generative model based on Distribution of Sparse Codes (DSC).
- A hybrid target model built on binary Classifier weighted Distribution of Sparse Codes of object parts (CWDSC).
- Novel objective functions based on the models.
- Sparse representation based trackers with *FRGA* and motion information based re-ranking for localization.

This chapter is organized as follows. Section 4.2 describes the approaches that are relevant to this work. Section 4.3 describes the dictionary learning procedure. Section 4.4 describes the generative object model from sparse code coefficients. Section 4.5 describes the hybrid object model. Section 4.6 describes the localization and motion inforation based re-ranking. Section 4.7 discusses the experiments conducted and the results obtained. Section 4.8 summarizes the present work and sketches possible future extensions.

4.2 Related Work

Sparse representation of signals has two important components, viz. dictionary learning and sparse coding. The dictionary can be predefined or learned from data. During the sparse coding stage, coefficients of a signal is computed with respect to the learned dictionary as described in subsection 4.2.1. Different techniques exist in literature to compute a dictionary like KSVD [258], discriminative dictionary learning [259], K-selection [72] etc.. Learned dictionaries are more popular in literature for sparse coding due to their expressive ability [271]. Sparse codes are computed using several algorithms like Orthogonal Matching Pursuit [272], Focal Underdetermined System Solver [273] among others [261, 262, 274] in literature. Mei *et al.* [23] was the fist to formulate tracking as a ℓ_1 minimization problem following which many modifications were proposed in literature. More on this

is explained in subsection 4.2.2. Usefulness of sparse codes in modeling an object is explained in subsection 4.2.3.

4.2.1 Sparse Representation

Let $\mathbf{p} \in \mathbb{R}^{n_\mathbf{p}}$ be a patch vector, which can be approximated as a linear combination of a few atoms from a dictionary $\mathbf{D} \in \mathbb{R}^{n_\mathbf{p} \times n_a}$ such that

$$\mathfrak{p} = \mathbf{D}\boldsymbol{\gamma} \tag{4.1}$$

where, $\mathbf{D}$ is the dictionary with n_a atoms given as $\mathbf{D} = [\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_{n_a}]$, $\|\hat{\mathbf{a}}\| = 1$ and $\boldsymbol{\gamma} \in \mathbb{R}^{n_a}$ is the sparse code vector. For computing sparse codes, the dictionary $\mathbf{D}$ as well as an algorithm to compute the sparse code are required. Traditional dictionaries like discrete cosine transform [275], wavelets [276], curvelets [277] and others exist in the literature. However, over complete dictionaries became popular due to their expressive ability [271]. More recent techniques follow learning the dictionary from data as they provide optimal atoms when compared to predefined ones. One such dictionary learning technique is the K-SVD algorithm [258].

Let $\mathbb{X}^+ = \{\mathbf{p}_i, i = 1, \ldots, n_{\mathbb{X}^+}\}, \mathbb{X} \in \mathbb{R}^{n_\mathbf{p} \times n_{\mathbb{X}^+}}$ be the set of all patch vectors extracted from object bounding box annotated region $\mathbf{bb}_0^+$ in frame $\mathcal{I}_0$. A dictionary is learned on these patch vectors with K-SVD algorithm by solving the following constrained optimization problem.

$$\min_{\mathbf{D},\boldsymbol{\Gamma}^+} \|\mathbb{X}^+ - \mathbf{D}\boldsymbol{\Gamma}^+\|_2^2 \; s.t. \; \|\boldsymbol{\Gamma}_l^+\|_0 \leq \eta_s \; \forall \; l = 1, \ldots, n_{\mathbb{X}^+} \tag{4.2}$$

where, $\boldsymbol{\Gamma}^+ = [\boldsymbol{\gamma}_1^+, \ldots, \boldsymbol{\gamma}_{n_{\mathbb{X}^+}}^+]^T (\boldsymbol{\Gamma}^+ \in \mathbb{R}^{n_a \times n_{\mathbb{X}^+}})$ is the matrix of sparse code vectors and η_s is the sparsity constraint. It iterates between dictionary update and sparse coding step. Given the dictionary $\mathbf{D}$, sparse code vectors $\boldsymbol{\Gamma}^+$ can be computed with Orthogonal Matching Pursuit (OMP) algorithm [263]. OMP is a greedy algorithm that alternates between sparse coefficient computation and residual calculation. An atom that is highly similar to the signal is selected while computing the coefficient in each iteration. Mei *et al.* [23] proposed sparse coding in a ℓ_1 optimization framework with a dictionary generated from object templates and its perturbations for tracking.

4.2.2 L1 Tracker

Mei *et al.* [23] was the first to use sparse code representation based object modeling for tracking in ℓ_1 minimization framework. Tracking was formulated as a sparse code approximation problem

in template subspace which was defined by the dictionary $\mathbf{D}$. The target object region in image $\mathcal{I}_0$ within bounding rectangle $\mathbf{bb}_0^+$ was vectorized to create a target template of size $\mathbf{p} \in \mathbb{R}^{n_\mathbf{p}}$. A number of object deformations are applied to an original target template in the form of local intensity changes to generate additional templates. A total of $n_\mathbb{T}$ templates are formed from this process. Additionally two trivial template sets are incorporated as $\mathbf{I}$ and $-\mathbf{I}$. Here $\mathbf{I} \in \mathbb{R}^{n_Q \times n_Q}$ is the identity matrix. A three part overcomplete dictionary consisting of target templates, positive and negative trivial templates is defined as follows.

$$\mathbf{D} = [\mathbf{D}^\mathbb{T} | \mathbf{I} | - \mathbf{I}] \tag{4.3}$$

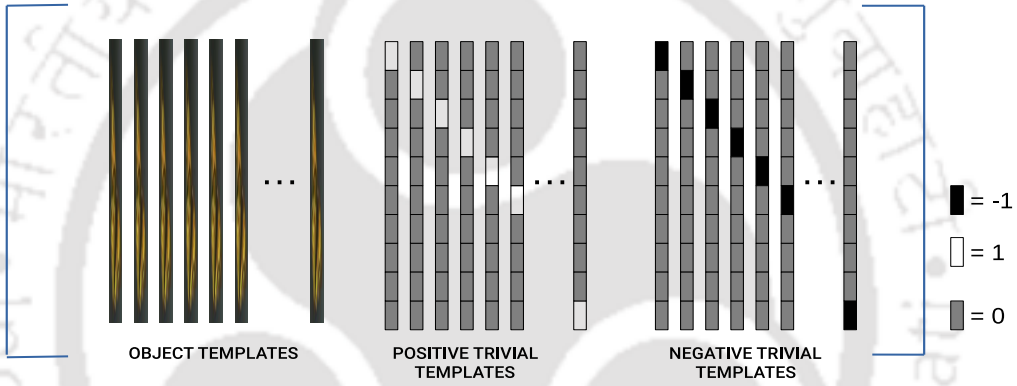

Figure 4.1: Dictionary consisting of object templates and trivial templates

where, $\mathbf{D}^\mathbb{T} = [\mathbf{p}_1^\mathbb{T}, \ldots, \mathbf{p}_{n_\mathbb{T}}^\mathbb{T}]$ are the set of vectorized target template variants. Generated dictionary is illustrated in Figure 4.1. During tracking, candidate region proposals are generated on search image $\mathcal{I}_t^S \in \mathcal{I}_t$. Let the set of all such proposals be $\mathbb{B}_t^+ = \{\mathbf{bb}_{ti}^+; i = 1, \ldots, n_\mathbb{B}\}$. The candidate template is extracted from $\mathcal{I}_t^S$ with $\mathbf{bb}_{ti}^+, (i = 1, \ldots, n_\mathbb{B})$ and is scaled to the size of $\mathbf{bb}_0^+$ and vectorized to form the set $\mathbb{X}_t^+ = \{\mathbf{p}_{t1}, \ldots, \mathbf{p}_{tj}, \ldots, \mathbf{p}_{tn_\mathbb{B}}\}$. The following optimization problem is solved for each patch vector $\mathbf{p}_{ti}$ to obtain the optimal sparse code vector $\boldsymbol{\gamma}_{ti}$

During tracking, any candidate region defined by the i^{th} bounding box $\mathbf{bb}_{ti}^+$ can be used to extract candidate patch $\mathbf{p}_{ti}$ from search image $\mathcal{I}_t^S$ and can be represented as a linear combination of atoms from $\mathbf{D}$ by solving the following.

$$\min \|\mathbf{D}\boldsymbol{\gamma}_{ti} - \mathbf{p}_{ti}\|_2^2 + \lambda \|\boldsymbol{\gamma}_{ti}\|_1 \tag{4.4}$$

where, λ is the regularization coefficient. This is solved as ℓ_1 regularizedleast squares problem [23].

The patch vector $\mathbf{p}_{ti}$ can be represented as follows

$$\mathbf{p}_{ti} = \mathbf{D}\boldsymbol{\gamma}_{ti} + \epsilon \tag{4.5}$$

where, ϵ is the reconstruction error, $\boldsymbol{\gamma}_{ti} = [\boldsymbol{\gamma}_{ti}^{\mathbb{T}}\ \boldsymbol{\gamma}_{ti}^{+}\ \boldsymbol{\gamma}_{ti}^{-}]$ and $\boldsymbol{\gamma}_{ti}^{\mathbb{T}}$, $\boldsymbol{\gamma}_{ti}^{+}$, $\boldsymbol{\gamma}_{ti}^{-}$ are the sparse coefficient vectors corresponding to the template, positive and negative trivial templates of the dictionary $\mathbf{D}$. Best candidate from the region proposal is the one that has the minimum residue r_{ti} computed as follows

$$r_{ti} = \|\mathbf{p}_{ti} - \mathbf{D}^{\mathbb{T}}\boldsymbol{\gamma}_{ti}^{\mathbb{T}}\|_2^2 \tag{4.6}$$

Mei *et al.* [23] used particle filter approach for localization. The best candidate box $\overset{\star}{\mathbf{bb}}_t^{+}$ is selected as

$$\overset{\star}{\mathbf{bb}}_t^{+} = \mathbf{bb}_{ti^\star}^{+} \ \ s.t. \ \ i^\star = \underset{i=1,\ldots,n_{\mathbb{B}}}{\arg\min} \|\mathbf{p}_{ti} - \mathbf{D}^{\mathbb{T}}\boldsymbol{\gamma}_{ti}^{\mathbb{T}}\|_2^2 \tag{4.7}$$

A good candidate can be represented with few templates resulting in a sparse solution with limited number of trivial coefficients. Noise and occlusion is captured by the trivial templates and are thus discarded while computing reconstruction error. Candidate in a background region will ideally select more atoms during the minimization stage. A candidate with the smallest projection error is identified as the tracker prediction $\overset{\star}{\mathbf{bb}}_t^{+}$. Localization is performed with particle filter during which sparse codes are computed for the particles.

4.2.3 Sparse coefficient based object model

Sparse coding is a popular way of learning feature representation [258]. Sparse codes are used to model object in vision related tasks [23, 256]. Ren *et al.* [256] proposed an object model based on sparse coefficients named Histogram of Sparse Codes (HSC). It is robust against appearance and illumination variations [256]. Sparse codes of patches help in capturing local information of the object. Per-pixel position sparse codes are computed for extracted patch and are then aggregated to form HSC. Ren *et al.* [256] highlighted that dictionary-based features can outperform popular HOG for general object detection. Sparse representation of patches in $\mathbb{X}^+$ are jointly learned by minimizing the reconstruction error over a dictionary $\mathbf{D}$ as given below.

$$\underset{\mathbf{D}\mathbb{X}^+}{\min} \|\mathbb{X}^+ - \mathbf{D}\boldsymbol{\Gamma}^+\|_F^2 \ s.t. \ \forall \ \|\boldsymbol{\gamma}_l^+\|_0 \leq \eta_s \ , l = 1, \ldots, n_{\mathbb{X}^+} \tag{4.8}$$

Here, η_s is the sparsity of signal, $\boldsymbol{\Gamma}^+$ is the sparse code matrix computed.

Object template $\mathcal{I}^Q$ is divided into 8×8 cells in a sliding window framework for the feature computation of each cell. All patches are extracted from a cell. Let γ_l^+ be the sparse code vector of l^{th} patch. For each $\gamma_l^+[j] \in \gamma_l^+ \; s.t. \; \gamma_l^+[j] \neq 0$, a soft binning strategy is followed to assign the absolute value $|\gamma_l^+[j]|$ to one of its spatially surrounding cells. This results in a feature vector on each cell averaging codes in a 16×16 neighborhood. Normalization and power transform is applied to the feature vector to define HSC. This is illustrated in Figure 4.2. In this work a similar but simpler formulation is used for object modeling (See section 4.4) from the sparse code coefficients. Dictionary learning approach followed in this work is explained next.

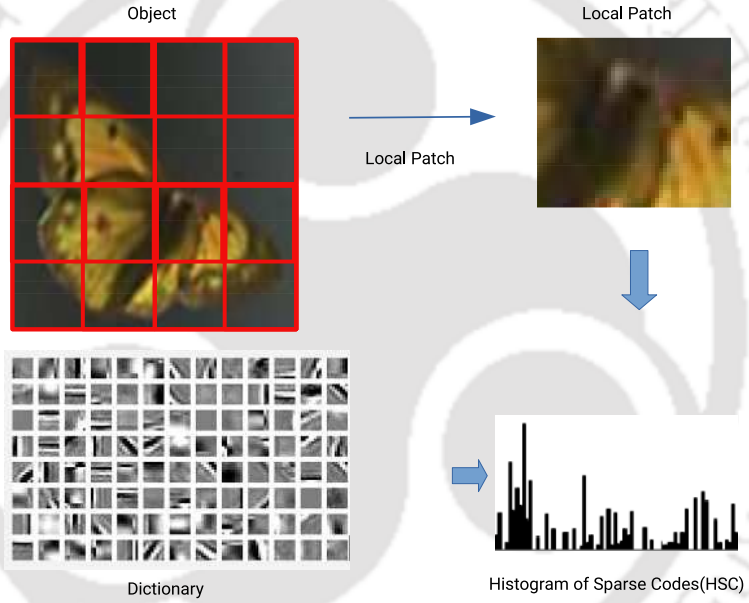

Figure 4.2: Sparse codes vectors of patches extracted from object region is computed with dictionary and are then used to compute Histogram of Sparse Codes (HSC).

4.3 Dictionary Learning

A single dictionary learned from only object patches might provide good reconstruction but poor recognition against background clutter. Accordingly, this work learns a dictionary with both object and its immediate neighbourhood (context) for object modeling.

Dictionary is learned in the first frame $\mathcal{I}_0$ with patches drawn from both object (target) and its immediate background. Let $\mathbf{bb}_0^+$ be the minimum bounding rectangle of the target object and $\mathbf{bb}_0^\pm$ be

the concentric and bigger rectangle. A region consisting of only background is then defined as follows

$$\mathbf{bb}_0^- = \mathbf{bb}_0^{\pm} - \mathbf{bb}_0^+ \tag{4.9}$$

Patches are sampled from two concentric rectangles $\mathbf{bb}_0^+$ and $\mathbf{bb}_0^-$. The extracted patches are first vectorized and then magnitude normalized (using $l2$-norm). Let $n_{\mathbb{X}+}$ and $n_{\mathbb{X}-}$ be the total number of patches that are extracted from $\mathbf{bb}_0^+$ and $\mathbf{bb}_0^-$ respectively. Next, they are arranged to construct the input data matrices $\mathbb{X}_0^+ \in \mathbb{R}^{n_\mathfrak{p} \times n_{\mathbb{X}+}}$ and $\mathbb{X}_0^- \in \mathbb{R}^{n_\mathfrak{p} \times n_{\mathbb{X}-}}$ of respective object (positive) and background (negative) classes. Here, $n_\mathfrak{p}$ is the dimension of patch vector.

Patch vectors in $\mathbb{X}_0^+$ and $\mathbb{X}_0^-$ are separately clustered to identify unique patterns in foreground and background. Here, hierarchical agglomerative clustering [278] is performed using cosine distance as the metric. Let n_{K+} and n_{K-} be the respective number of clusters obtained from $\mathbb{X}_0^+$ and $\mathbb{X}_0^-$. Clustering does not ensure uniform distribution of data across clusters. Also, the data is not uniformly distributed in space. Thus, all $n_{K+} + n_{K-}$ clusters are oversampled (using SMOTE algorithm [279]) to have uniform number of vectors $n_{c\mathbb{X}}$ (number of vectors in largest cluster). After this, every cluster has a total of $n_{c\mathbb{X}}$ patch vectors. This ensures the availability of sufficient number of vectors in each cluster and counter data imbalance problem.

Let $\mathbb{X}_{0c}^+ \in \mathbb{R}^{n_\mathfrak{p} \times n_{c\mathbb{X}}}$ be the the data matrix of c^{th} cluster in the foreground. The dictionary $\mathbf{D}_c^+$ with n_k atoms ($\mathbf{D}_c^+ \in \mathbb{R}^{n_\mathfrak{p} \times n_k}$, $c = 1, \ldots, n_{K+}$) is learned from the vectors in c^{th} cluster using K-SVD algorithm. Thus, atoms in $\mathbf{D}_c^+$ provide a representation for vectors in the c^{th} cluster. The dictionary $\mathbf{D}_c^+$ is learned by solving the following optimization problem.

$$\min_{\mathbf{D}_c^+, \mathbf{\Gamma}_{0c}^+} \|\mathbb{X}_{0c}^+ - \mathbf{D}_c^+ \mathbf{\Gamma}_{0c}^+\|_2^2 \; s.t. \; \|\boldsymbol{\gamma}_{lc}^+\|_0 \leq \eta_s \tag{4.10}$$

Here, η_s is the sparsity parameter, $\boldsymbol{\gamma}_{lc}^+ \in \mathbb{R}^{n_k}$ is the sparse code vector of l^{th} patch vector and $\mathbf{\Gamma}_{0c}^+ \in \mathbb{R}^{n_k \times n_{c\mathbb{X}}}$. The sparse code vectors are stacked to obtain sparse code matrix $\mathbf{\Gamma}_{0c}^+ \in \mathbb{R}^{n_k \times n_{c\mathbb{X}}}$. Atoms learned from foreground clusters are stacked to form the foreground dictionary $\mathbf{D}^+ = [\mathbf{D}_c^+], c = 1 \ldots, n_{K+}$ ($\mathbf{D}^+ \in \mathbb{R}^{n_\mathfrak{p} \times n_{a+}}$, $n^{a+} = n_k \times n_{K+}$).

$$\mathbf{D}^+ = [\mathbf{D}_1^+, \ldots, \mathbf{D}_c^+, \ldots, \mathbf{D}_{n_{K+}}^+] \tag{4.11}$$

Similarly, the background dictionary $\mathbf{D}^-$ ($\mathbf{D}^- \in \mathbb{R}^{n_\mathfrak{p} \times n_{a-}}$) is also computed using clustering followed by K-SVD. Let $\mathbb{X}_{0c}^- \in \mathbb{R}^{n_\mathfrak{p} \times n_{c\mathbb{X}}}$ be the the data matrix of c^{th} cluster in the background. The dictionary

$\mathbf{D}_c^-$ with n_k atoms ($\mathbf{D}_c^- \in \mathbb{R}^{n_\mathfrak{p} \times n_k}, c = 1, \ldots n_{K-}$) is learned from the vectors in c^{th} cluster using K-SVD algorithm. Thus, atoms in $\mathbf{D}_c^-$ provide a representation for vectors in the c^{th} cluster. The dictionary $\mathbf{D}_c^-$ is learned by solving the following optimization problem.

$$\min_{\mathbf{D}_c^-, \mathbf{\Gamma}_{0c}^-} \|\mathbb{X}_{0c}^- - \mathbf{D}_c^- \mathbf{\Gamma}_{0l}^-\|_2^2 \; s.t. \; \|\boldsymbol{\gamma}_{lc}^-\|_0 \leq \eta_s \tag{4.12}$$

Here, η_s is the sparsity parameter, $\boldsymbol{\gamma}_{lc}^- \in \mathbb{R}^{n_k}$ is the sparse code vector of l^{th} patch vector and $\mathbf{\Gamma}_{0c}^- \in \mathbb{R}^{n_k \times n_{c\mathbb{X}}}$. The sparse code vectors are stacked to obtain sparse code matrix $\mathbf{\Gamma}_{0c}^- \in \mathbb{R}^{n_k \times n_{c\mathbb{X}}}$. Atoms learned from background clusters are stacked to form the background dictionary $\mathbf{D}^- = [\mathbf{D}_c^-], c = 1, \ldots, n_{K-}$ ($\mathbf{D}^- \in \mathbb{R}^{n_\mathfrak{p} \times n_{a-}}$, $n^{a-} = n_k \times n_{K-}$).

$$\mathbf{D}^- = [\mathbf{D}_1^-, \ldots, \mathbf{D}_c^-, \ldots, \mathbf{D}_{n_{K-}}^-] \tag{4.13}$$

An overcomplete dictionary consisting of foreground and background atoms is constructed as $\mathbf{D} = [\mathbf{D}^+ \; |\mathbf{D}^-] \in \mathbb{R}^{n_\mathfrak{p} \times n_a}$. Here, $\mathbf{D} \in \mathbb{R}^{n_\mathfrak{p} \times n_a}$ and $n_a = n_{a+} + n_{a-}$. Dictionary learning procedure is illustrated in Figure 4.3. Learned dictionary can represent both foreground (target object) and scene background. It is then used to the define object model as explained next.

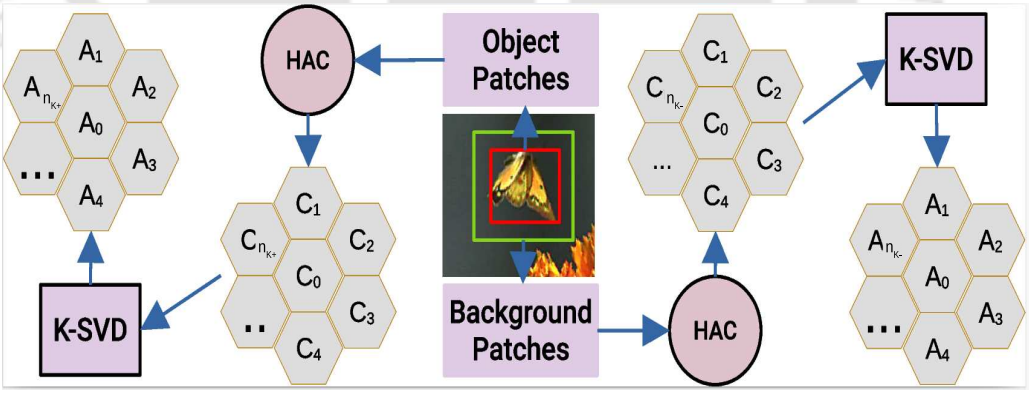

Figure 4.3: Patches extracted from foreground and background regions are vectorized and magnitude normalized. These are further grouped using hierarchical clustering. K-SVD performed on cluster members provides us with different atom sets. These atom sets obtained from background and object clusters are combined to form a single dictionary.

4.4 Distribution of Sparse Codes

In this approach, explicit use of a target model is avoided and only candidate models are used in the localization stage. Consider a search image $\mathcal{I}_t^S \in \mathcal{I}_t$. Candidate region proposals are generated

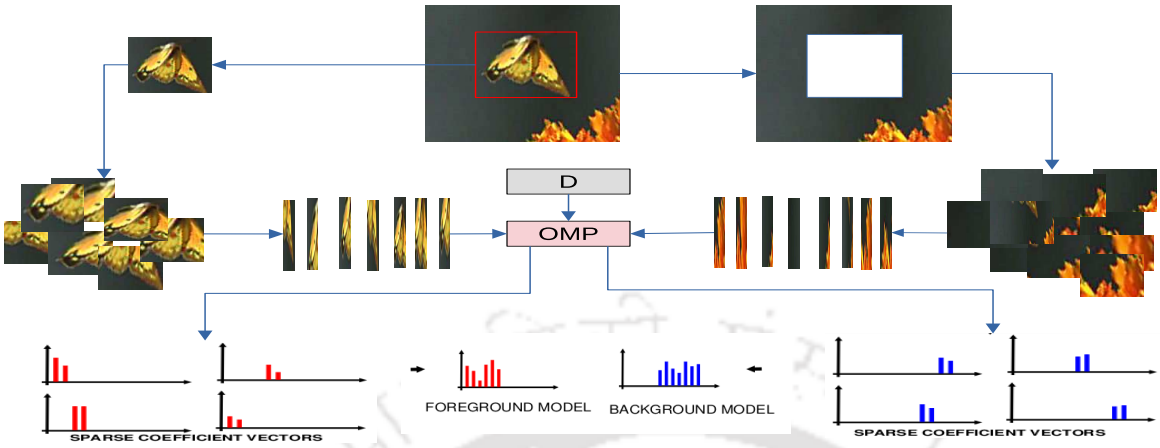

Figure 4.4: Vectorized and magnitude normalized patches drawn from foreground and background region are subjected to OMP (with learned dictionary) for computation of their sparse codes. A distribution of sparse codes obtained from foreground and background regions define the candidate models.

within $\mathcal{I}_t^S$ to form the set $\mathbb{B}_t^+ = \{\mathbf{bb}_{ti}^+, i = 1\ldots, n_{\mathbb{B}}\}$. Concentric bounding rectangle set $\mathbb{B}_t^\pm$ and $\mathbb{B}_t^-$ are generated as follows

$$\mathbf{bb}_{ti}^- = \mathbf{bb}_{ti}^\pm - \mathbf{bb}_{ti}^+ \tag{4.14}$$

Here, $\mathbf{bb}_{ti}^\pm \in \mathbb{B}_t^\pm$ and $\mathbf{bb}_{ti}^- \in \mathbb{B}_t^-$.

Each candidate region proposal is resized and reoriented to the size of $\mathbf{bb}_0^+$. Patches extracted from $\mathbf{bb}_{ti}^+$ are vectorized and magnitude normalized to form $n_\mathfrak{p}$ dimensional patch vector. A total of $n_{\mathbb{X}+}$ patch vectors are stacked to form the candidate patch vector matrix $\mathbb{X}_{ti}^+ \in \mathbb{R}^{n_\mathfrak{p} \times n_{\mathbb{X}+}}$. Sparse code vectors $\boldsymbol{\gamma}_{ti}^+, i = 1, \ldots, n_{\mathbb{X}+}$ are learned with the dictionary $\mathbf{D}$ (equation4.13) through the following constrained optimization problem.

$$\min_{\boldsymbol{\Gamma}_{ti}^+} \|\mathbb{X}_{ti}^+ - \mathbf{D}\boldsymbol{\Gamma}_{ti}^+\|_2^2 \; s.t. \; \|\boldsymbol{\gamma}_{til}^+\|_0 \leq \eta_s \tag{4.15}$$

Here, $\boldsymbol{\Gamma}_{ti}^+ \in \mathbb{R}^{n_a \times n_{\mathbb{X}+}}$ is the sparse code matrix obtained by stacking $\boldsymbol{\gamma}_{til}^+, l = 1, \ldots, n_{\mathbb{X}+}; i = 1, \ldots, n_{\mathbb{B}}$ and η_s is the sparsity parameter. Orthogonal Matching Pursuit (OMP) [280]) is used to compute the spare code vectors.

Similarly patches are extracted from $\mathbf{bb}_{ti}^-$ (immediate background of $\mathbf{bb}_{ti}^+$). These patches are vectorized and magnitude normalized to obtain a total of $n_{\mathbb{X}}-$ patch vectors. They are stacked to form the data matrix $\mathbb{X}_{ti}^- \in \mathbb{R}^{n_\mathfrak{p} \times n_{\mathbb{X}-}}$

$$\min_{\boldsymbol{\Gamma}_{ti}^-} \|\mathbb{X}_{ti}^- - \mathbf{D}\boldsymbol{\Gamma}_{ti}^-\|_2^2 \; s.t. \; \|\boldsymbol{\gamma}_{til}^-\|_0 \leq \eta_s \tag{4.16}$$

Here, $\mathbf{\Gamma}_{ti}^{-} \in \mathbb{R}^{n_a \times n_{\mathbb{X}-}}$ is the sparse code matrix obtained by stacking $\boldsymbol{\gamma}_{til}^{-}, l = 1, \ldots, n_{\mathbb{X}-}; i = 1, \ldots, n_{\mathbb{B}}$.

For an optimal candidate region (region of the target object) it is expected that patches from $\mathbf{bb}_{ti}^{+}$ will choose atoms from $\mathbf{D}^{+}$ and $\mathbf{bb}_{ti}^{-}$ will choose atoms from $\mathbf{D}^{-}$ during the sparse coding stage. This makes the sparse codes $\mathbf{\Gamma}_{ti}^{+}$ and $\mathbf{\Gamma}_{ti}^{-}$ to have no overlap between them. This property is used in the localization phase (See section 4.6).

Candidate model is computed as the distribution of sparse codes and is designed to have a foreground and a background component. Foreground component $\mathbf{m}_{ti}^{\mathbb{C}+}$ is computed from the patch vectors $\mathbb{X}_{ti}^{+}$ and is given by

$$\mathbf{m}_{ti}^{\mathbb{C}+}[j] = L_{ti}^{+} \sum_{l=1}^{n_{\mathbb{X}+}} \left| \boldsymbol{\gamma}_{til}^{+}[j] \right|, \; j = 1, \ldots, n_a, \; l = 1, \ldots, n_{\mathbb{X}+} \tag{4.17}$$

where, L_{ti}^{+} is the normalization coefficient such that $\sum_{j=1}^{n_a} \mathbf{m}_{ti}^{\mathbb{C}+}[j] = 1$

Similarly, for the background component, patches extracted from $\mathbf{bb}_{ti}^{-}$ are vectorized and magnitude normalized to construct $\mathbb{X}_{ti}^{-} \in \mathbb{R}^{n_{\mathbb{p}} \times n_{\mathbb{X}-}}$. $\mathbf{m}_{ti}^{\mathbb{C}-}$ represents the background component of candidate model and is computed from $\mathbb{X}_{ti}^{-}$ using dictionary $\mathbf{D}$ as follows.

$$\mathbf{m}_{ti}^{\mathbb{C}-}[j] = L_{ti}^{-} \sum_{l=1}^{n_{\mathbb{X}-}} \left| \boldsymbol{\gamma}^{-}{}_{til}[j] \right|, \; j = 1, \ldots, n_a, \; l = 1, \ldots, n_{\mathbb{X}-} \tag{4.18}$$

where, L_{ti}^{-} is the normalization coefficient such that $\sum_{j=1}^{n_a} \mathbf{m}_{ti}^{\mathbb{C}-}[j] = 1$. These models are used in the localization stage where an objective function is defined on them.

A potential foreground region $\mathbf{bb}_{ti}^{+}$ should mostly contain object patches. Thus, the first n_{a+} atoms in $\mathbf{D}$ (i.e. from $\mathbf{D}^{+}$) should mostly contribute to $\mathbf{m}_{ti}^{\mathbb{C}+}$. We define the extent of presence of target object in $\mathbf{bb}_{ti}^{+}$ as the net contribution from object atoms in $\mathbf{m}_{ti}^{\mathbb{C}+}$ and is given by

$$\rho_{ti}^{+} = \frac{\sum_{j=1}^{n_{a+}} \mathbf{m}_{ti}^{\mathbb{C}+}[j]}{\sum_{j=1}^{n_a} \mathbf{m}_{ti}^{\mathbb{C}+}[j]} \tag{4.19}$$

A proposal bounding box $\mathbf{bb}_{ti}^{+}$ containing only object information gives high value of ρ^{+} in two cases. First one refers to perfect target localization. However, in the second case, the region proposal $\mathbf{bb}_{ti}^{+}$ might be smaller than the actual object size due to scale implosion. This indicates the shortcomings of using only foreground model. The scale implosion can be arrested by testing the presence of background patches in immediate neighborhood of $\mathbf{bb}_{ti}^{+}$ i.e. in $\mathbf{bb}_{ti}^{-}$. The extent of presence of

background patches in $\mathbf{bb}_{ti}^{-}$ can be similarly measured by the net contribution of background atoms (last n_{a-} atoms of $\mathbf{D}$, i.e. $\mathbf{D}^{-}$) in $\mathbf{m}_{ti}^{\mathbb{C}-}$. This can be defined as follows

$$\rho_{ti}^{-} = \frac{\sum_{j=n_{a+}+1}^{n_a} \mathbf{m}_{ti}^{\mathbb{C}-}[j]}{\sum_{j=1}^{n_a} \mathbf{m}_{ti}^{\mathbb{C}-}[j]} \tag{4.20}$$

An objective function defined as $\mathcal{J}_{ti} = \rho_{ti}^{+} \times \rho_{ti}^{-}$ will only partially solve the problem. In cases of scale explosion, $\mathbf{bb}_{ti}^{+}$ may fully contain the target while being larger in size. Here, both ρ_{ti}^{-} and ρ^{+} might have high values. A proper observation likelihood should (a) maximize the extent of presence of target object within $\mathbf{bb}_{ti}^{+}$, (b) maximize the presence of background in $\mathbf{bb}_{ti}^{-}$ and (c) minimize the presence of background in $\mathbf{bb}_{ti}^{+}$. The measures ρ_{ti}^{+} and ρ_{ti}^{-} handles the first two conditions. The third criteria is quantified as the Bhattacharyya distance between the foreground and background distributions. This measure (ρ_{ti}^{d}) will have higher value if $\mathbf{m}_{ti}^{\mathbb{C}+}$ and $\mathbf{m}_{ti}^{\mathbb{C}-}$ are non-overlapping. This is only possible when the third condition is satisfied.

$$\rho_{ti}^{d} = 1 - \sum_{j=1}^{n_a} \sqrt{\mathbf{m}_{ti}^{\mathbb{C}+}[j]} \sqrt{\mathbf{m}_{ti}^{\mathbb{C}-}[j]} \tag{4.21}$$

It can be observed that the optimal candidate region proposal $\overset{\star}{\mathbf{bb}}_{ti}^{+} \in \mathbb{B}_{t}^{+}$ should be the one that simultaneously maximizes $\rho_{ti}^{+}, \rho_{ti}^{-}$ and $\rho_{ti}^{\pm}$

$$\mathcal{J}_{ti}^{dsc}(\mathbf{bb}_{ti}^{\pm}) = \rho_{ti}^{\pm} = \rho_{ti}^{+} \times \rho_{ti}^{-} \times \rho_{ti}^{d} \tag{4.22}$$

This objective function is optimized during the localization stage (section 4.6) to choose the optimal region proposal $\overset{\star}{\mathbf{bb}}_{t}^{+}$ that maximizes $\mathcal{J}_{dsc}(.)$

Experiments were conducted for visualizing the localization capabilities of the individual measures. A dictionary was computed on the first frame of the BALL sequence from $VOT2014$ dataset. Candidate regions were extracted from the second frame as sliding windows (without scaling and rotation). The proposed measures were computed on these candidate regions. The measurement values over the entire frame (excluding boundary pixels) are shown in Figures 4.5(b)-(e). This visualization illustrates better localization power of the proposed objective function compared to the individual ones. Several tracking approaches have employed discriminative models to differentiate target object from background. Classifiers are generally used for this [51, 104, 281]. Accordingly an object model based on sparse code coefficients and discriminative classifier is explored next.

(a) (b) (c) (d) (e)

Figure 4.5: (a) First frame from BALL sequence in VOT2014 dataset. Observation likelihood value at every pixel location (excluding boundary) on the second frame computed with proposed measures (b) ρ_+ (equation 4.19 (c) ρ_- (equation 4.20)) (d) ρ_d (equation 4.21) and (e) $\mathcal{J}$ (equation 4.22)

4.5 Classifier weighted Distribution of Sparse Codes

Dictionary is learned from the clustered patches of the foreground region $\mathbf{bb}_0^+$ and background region $\mathbf{bb}_0^-$ as explained in section 4.3. An object to be tracked can be of any shape in general. Target object annotation the object region with a rectangular box $\mathbf{bb}_0^+$ might result in the presence of background patches in it. This could create atoms in the dictionary that are common between both object and background. Ideally, the sparse codes of patches from object should choose atoms from $\mathbf{D}^+$ and patches from background should select atoms from $\mathbf{D}^-$. The presence of overlapping information between foreground and background can affect object modeling. To overcome this, a two stage process is adopted. Model is computed with sparse code coefficients of object patch vectors. It is them modified with the weights defined from a binary classifier. It is a common practice in object modeling to give importance to each of the object pixels. For example in [4] kernel weights are used for this. Pixels closer to object centroid are weighted more in comparison to the ones at the boundary region. However, this approach is inefficient as the objectiveness of pixels might not vary in that fashion. Some patches might be unique to the object, some might be common with the background and some are only present in background. To model an object, patches are to be given importance according to their chance of being present in the template image $\mathcal{I}^Q$ derived from $\mathcal{I}_0$ with the bounding box annotation $\mathbf{bb}_0^+$. Unique object patches must be given more importance compared to the ones common to both object and background. Background patches present in $\mathbf{bb}_0^+$should be given least importance while modeling an object. Accordingly, this proposal assigns weights to each patch according to their distinction with respect to the background. This is achieved by learning a binary classifier that assign weights to each pixel position after extracting patches around them. The classifier is learned on sparse code vectors of patches extracted from the object region $\mathbf{bb}_0^+$ and the

background region $\mathbf{bb}_0^-$. This is also similar to the object model correction applied in 3.2.2 where the pixel color intensities dominant in the background are nullified in the target model. A similar approach is adopted by using a learned binary classifier.

Patches extracted from $\mathbf{bb}_0^+$ are vectorized and magnitude normalized to form the matrix $\mathbb{X}_0^+ \in \mathbb{R}^{n_\mathfrak{p} \times n_{\mathbb{X}+}}$. Here, $n_{\mathbb{X}+}$ is the total number of patches extracted from $\mathbf{bb}_0^+$. Sparse codes corresponding to the patch vector are computed as follows

$$\min_{\mathbf{\Gamma}_0^+} \|\mathbb{X}_0^+ - \mathbf{D}\mathbf{\Gamma}_0^+\|_2^2 \; s.t. \; \|\mathbf{\Gamma}_{0l}^+\|_0 \leq \eta_s \tag{4.23}$$

Here, $\mathbf{\Gamma}_0^+ \in \mathbb{R}^{n_a \times n_{\mathbb{X}+}}$ is the sparse code matrix obtained by stacking $\boldsymbol{\gamma}_{0l}^+, l = 1, \ldots, n_{\mathbb{X}+}$, $\boldsymbol{\gamma}_{0l}^+$ is the sparse code vector of the l^{th} patch in $\mathbb{X}_0^+$ and η_s is the sparsity parameter. Orthogonal Matching Pursuit (OMP) [280]) is used to compute the spare code vectors. Similarly, patches extracted from $\mathbf{bb}_0^-$ are vectorized and magnitude normalized to form the matrix $\mathbb{X}_0^- \in \mathbb{R}^{n_\mathfrak{p} \times n_{\mathbb{X}-}}$. Sparse codes corresponding to the patch vectors are computed as follows

$$\min_{\mathbf{\Gamma}_0^-} \|\mathbb{X}_0^- - \mathbf{D}\mathbf{\Gamma}_0^-\|_2^2 \; s.t. \; \|\mathbf{\Gamma}_{0l}^-\|_0 \leq \eta_s \tag{4.24}$$

where, $\mathbf{\Gamma}_0^- \in \mathbb{R}^{n_a \times n_{\mathbb{X}-}}$ is the sparse code matrix obtained by stacking $\boldsymbol{\gamma}_{0l}^-, l = 1, \ldots, n_{\mathbb{X}-}$

A binary classifier is learned on the sparse code vectors of patches rather than on their intensity values. $\mathbf{\Gamma}_0^+$ and $\mathbf{\Gamma}_0^-$ are then used for training a binary Support Vector Machine (SVM) classifier. The learned classifier provides confidence scores to patches indicating their objectiveness value. These scores are used as weights to improve discrimination between patches belonging to foreground and background. A SVM classifier with linear kernel is learned on the sparse code representations of patch vectors as shown in Figure 4.6. The sparse coding stage takes the patch vectors into a higher dimensional space with few non-zero coefficients. Thus, a linear kernel is preferred over RBF kernel for the SVM classifier. The sparse code vector corresponding to an ambiguous patch (common to object and background) will lie closer to the classification boundary and will have a lower classification score compared to the object patches. This will improve the discrimination power of the object model against background and ambiguous patches.

This work follows a part-based object model following its wide use [15, 102, 282] in the tracking literature for countering occlusion. Object bounding box $\mathbf{bb}_0^+$ is first divided into n_r non-overlapping cells $\mathbb{BC}_0 = \{\mathbf{bc}_{0r} : \mathbf{bc}_{0r} \in \mathbb{R}^{w^c \times h^c)} r = 1, \ldots, n_r\}$, where w^c and h^c are the width and height of the

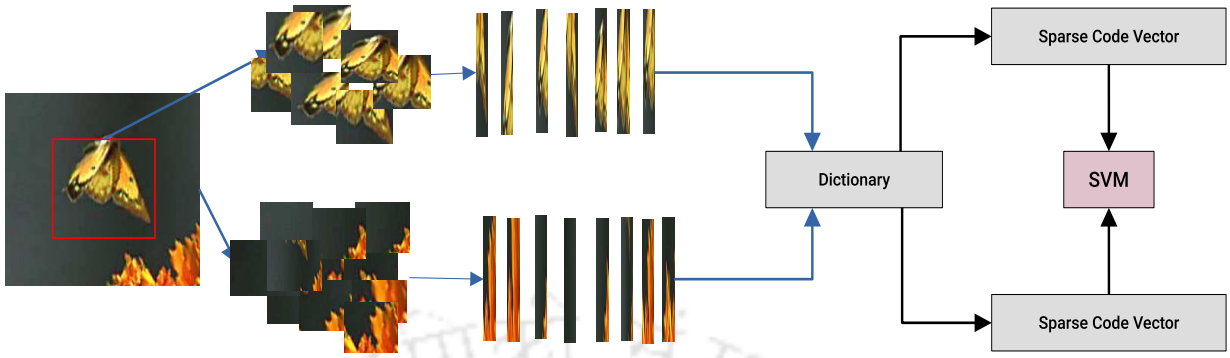

Figure 4.6: Training of the linear SVM classier on sparse code representations of object and background patches. $\mathbf{\Gamma}_0^+$ is the positive class data and $\mathbf{\Gamma}_0^-$ is the negative class data.

cell respectively. An object model is computed for each cell of the object. Sparse codes computed from patches in a cell are used to compute the model. The list of all cell models define the object model i.e. $\mathbf{M}_0^{\mathbb{T}} = [\mathbf{m}_{0r}^{\mathbb{T}+}]; r = 1, \ldots, n_r$. The sparse codes required for computing the object model are computed with equation 4.25.

$$\min_{\mathbf{\Gamma}_{0r}^+} \|\mathbb{X}_{0r}^+ - \mathbf{D}\mathbf{\Gamma}_{0r}^+\|_F^2 \; s.t. \; \|\mathbf{\Gamma}_{0rl}^+\|_0 \leq \eta_s \tag{4.25}$$

where, $\mathbf{\Gamma}_{0r}^+ = [\boldsymbol{\gamma}_{0r1}^+, \boldsymbol{\gamma}_{0r2}^+, \ldots, \boldsymbol{\gamma}_{0rn_{\mathbb{X}r}}^+]$ is the sparse code vector matrix of r^{th} cell obtained by stacking $\boldsymbol{\gamma}_{0rl}^+ \in \mathbb{R}^{n_a}, l = 1 \ldots, n_{\mathbb{X}r}$. $\boldsymbol{\gamma}_{0rl}^+$ is the sparse code vector of l^{th} patch of r^{th} cell. $\mathbb{X}_{0r}^+$ is the patch vector matrix of all the patches extracted from the r^{th} cell and $n_{\mathbb{X}r}$ is the total number of patches extracted. $\mathbf{D}$ is the foreground-background dictionary and η_s is the sparsity constraint. The distribution of sparse codes (DSC) of the cell is created from the sparse codes $\mathbf{\Gamma}_{0r}^+$ of its component patches as

$$\mathbf{m}_{0r}^{\mathbb{T}+}[j] = L_{0r}^{T+} \sum_{l=1}^{n_{\mathbb{X}r}} |\boldsymbol{\gamma}_{0lr}^+[j]| \; \omega(\boldsymbol{\gamma}_{0rl}^+), \; \boldsymbol{\gamma}_{0rl}^+ \in \mathbf{\Gamma}_{0r}^+ \tag{4.26}$$

Here, $\omega_(\boldsymbol{\gamma}_{0rl}^+)$ is the normalized classification score of the l^{th} patch in the r^{th} cell as patch as given by the classifier and $\sum_{l=1}^{n_{\mathbb{X}r}} \omega(\boldsymbol{\gamma}_{0rl}^+) = 1$. During localization, candidate region proposals are generate in a search image $\boldsymbol{\mathcal{I}}_t^S \in \boldsymbol{\mathcal{I}}_t$ which is given as $\mathbb{B}_t^+ = \{\mathbf{bb}_{ti}^+, i =, \ldots, n_{\mathbb{B}}\}$. Each candidate region is scaled to the size of $\mathbf{bb}_0^+$. Scaled candidate bounding box $\mathbf{bb}_{ti}^+$ is first divided into non-overlapping cells $\mathbb{BC}_{ti} = \{\mathbf{bc}_{tir} : r = 1, \ldots, n_r, \mathbf{bc}_{tir} \in \mathbb{R}^{w^c \times h^c)}\}$, where w^c and h^c are the width and height of the cell respectively. Sparse codes computed from patches in a cell are used to compute the candidate model for each cell. The list of all candidate cell models define the candidate model i.e. $\mathbf{M}_{ti}^{\mathbb{C}+} = [\mathbf{m}_{tir}^{\mathbb{C}+}]$ and $r = 1, \ldots, n_r$. The sparse codes required for the candidate model are computed with equation 4.27.

$$\min_{\mathbf{\Gamma}^{+}_{tir}} \|\mathbb{X}^{+}_{tir} - \mathbf{D}\mathbf{\Gamma}^{+}_{tir}\|^2_F \text{ s.t. } \|\mathbf{\Gamma}^{+}_{tirl}\|_0 \leq \eta_s \tag{4.27}$$

where $\mathbf{\Gamma}^{+}_{tir}$ is the sparse code matrix obtained by stacking $\boldsymbol{\gamma}^{+}_{tirl} \in \mathbb{R}^{n_a}, l = 1, \ldots, n_{\mathbb{X}r}$. $\boldsymbol{\gamma}^{+}_{tirl}$ is the sparse code vector of l^{th} patch of r^{th} cell in i^{th} candidate, $\mathbf{D}$ is the foreground-background dictionary and η_s is the sparsity constraint. The distribution of sparse codes (DSC) of the cell is created from the sparse codes $\mathbf{\Gamma}^{+}_{0r}$ of its component patches as follows.

$$\mathbf{m}^{\mathbb{C}+}_{tir}[j] = L^{\mathbb{C}+}_{tir} \sum_{i=1}^{n_{\mathbb{X}r}} |\boldsymbol{\gamma}^{+}_{tirl}[j]|\ \omega(\boldsymbol{\gamma}^{+}_{tirl}),\ \boldsymbol{\gamma}^{+}_{tirl} \in \mathbf{\Gamma}^{+}_{tir} \tag{4.28}$$

where, $\omega(\boldsymbol{\gamma}^{+}_{tirl})$ is the normalized classification score of the sparse code vector $\boldsymbol{\gamma}^{+}_{tirl}$ and $\sum_{l=1}^{n_{\mathbb{X}r}} \omega(\boldsymbol{\gamma}^{+}_{tirl}) = 1$, $n_{\mathbb{X}r}$ is the total number of patches in a cell, $\omega\boldsymbol{\gamma}_{tirl}$ is the classifier prediction probability that patch $\mathbf{p}_{tirl}$ belongs to the object and $L^{\mathbb{C}+}_{tir}$ is the normalization constant for the sparse coefficient distribution. The cell-wise distributions are collected together to form the object model $\mathbb{M}^{\mathbb{C}}_{ti}$ which is a list of classifier weighted distribution of sparse codes (CWDSC) given by

$$\mathbb{M}^{\mathbb{C}}_{ti} = [\ \mathbf{m}^{\mathbb{C}+}_{ti1}, \ldots, \mathbf{m}^{\mathbb{C}+}_{tir}, \ldots, \mathbf{m}^{\mathbb{C}+}_{tin_r}\ ] \tag{4.29}$$

Similarity between the model is used to define a objective function for localization step. This is explained next.

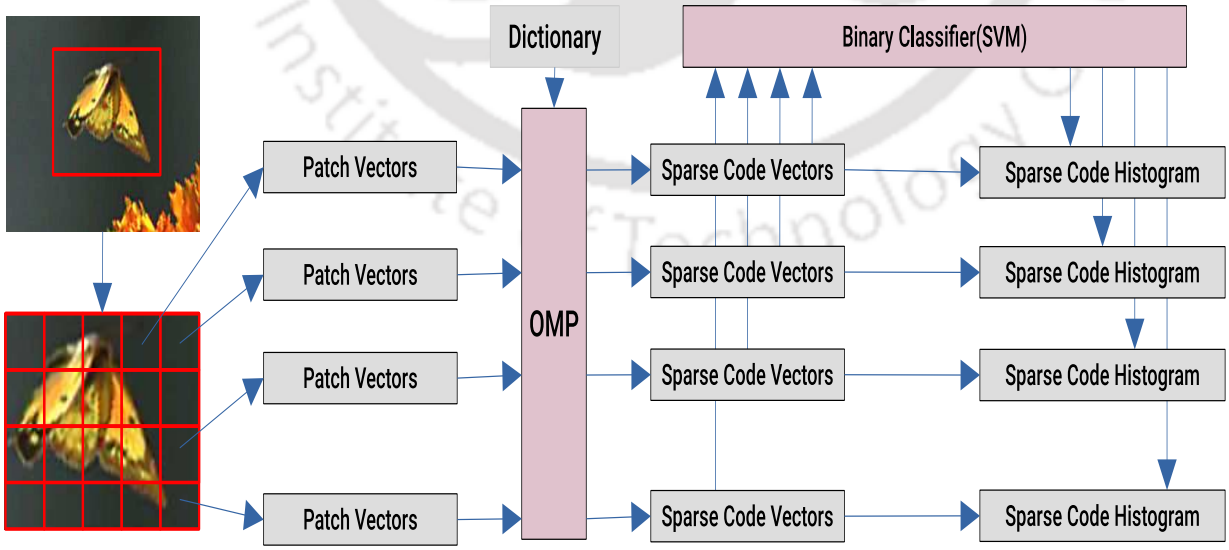

Figure 4.7: Object model as set of weighed sparse code distribution computed from non-overlapping cells of the object bounding box $\mathbf{bb}^{+}_{0}$. Sparse codes for the patches are computed using OMP (Orthogonal Matching Pursuit) with the learned dictionary. Sparse code vectors of the object patches and background patches are used for training the classifier. Classification score weighted sparse codes are then used to compute the distribution of each cell.

Objective function is defined based on the similarity between target and candidate models of each cell. Since classifier weights are used in the models, while performing similarity computation object patches will have more contribution compared to background patches. Overall measure is computed as the average similarity over the cells

$$\mathcal{J}_{ti} = \frac{1}{n_r}\sum_{r=1}^{n_r}\rho_{tir}^{+} = \frac{1}{n_r}\sum_{j=1}^{n_a}\sqrt{\mathbf{m}_{0r}^{\mathbb{T}+}[j] \times \mathbf{m}_{tir}^{\mathbb{C}+}[j]} \tag{4.30}$$

Target and candidate models of the r^{th} cell is given by $\mathbf{m}_{0r}^{\mathbb{T}+}$ and $\mathbf{m}_{tir}^{\mathbb{C}+}$ respectively, n_a is the dimension of the sparse code vector. The box $\overset{\star}{\mathbf{bb}}_t^{+}$ with highest average Bhattacharyya coefficient is selected as the optimal bounding box $\overset{\star}{\mathbf{bb}}_t^{+}$ in t^{th} frame.

4.6 Localization & Motion Correction

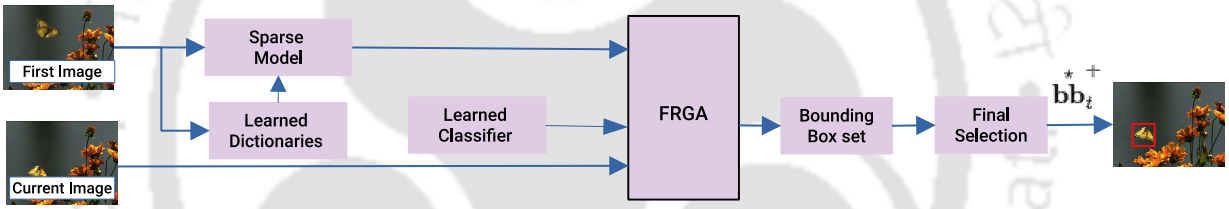

Figure 4.8: Block diagram of proposed approach with appearance component.

Localization is achieved with the *FRGA* 3.3.4 algorithm introduced in previous section 3.3.4 (Chapter3). Let $\mathbb{P}_t = \{\mathbf{z}_{ti}, i = 1, \ldots, n_{\mathbb{P}}\}$ be the population sampled on frame $\boldsymbol{\mathcal{I}}_t$. Let $\mathbf{z}_{ti} = \mathbf{bb}_{ti}^{+}$ represents i^{th} candidate region proposal in frame $\boldsymbol{\mathcal{I}}_t$. Objective function defined in equation 4.30 is used to define the brightness value of a firefly. For hybrid model, classifier weighted distribution of sparse codes (CWDSC) approach this is defined as

$$\mathcal{J}_{ti} = \frac{1}{n_r}\sum_{r=1}^{n_r}\rho_{tir}^{+} = \frac{1}{n_r}\sum_{j=1}^{n_a}\sqrt{\mathbf{m}_{0r}^{\mathbb{T}+}[j] \times \mathbf{m}_{tir}^{\mathbb{C}+}[j]} \tag{4.31}$$

For the generative model with distribution of sparse codes (DSC) based approach it is given as

$$\mathcal{J}^{dsc}(\mathbf{bb}_{ti}^{\pm}) = \rho_{ti}^{\pm} = \rho_{ti}^{+} \times \rho_{ti}^{-} \times \rho_{ti}^{d} \tag{4.32}$$

The functional block block diagram of the overall approach is as shown in Figure 4.8. Similar to the approach introduced in the previous section 3.4 (Chapter3) appearance features are combined with motion information computed with optical flow. Overall block diagram of the approach is shown in

Figure 4.9. Algorithm for the hybrid model based approach is presented in Algorithm 3 and generative model based approach is presented in Algorithm 2.

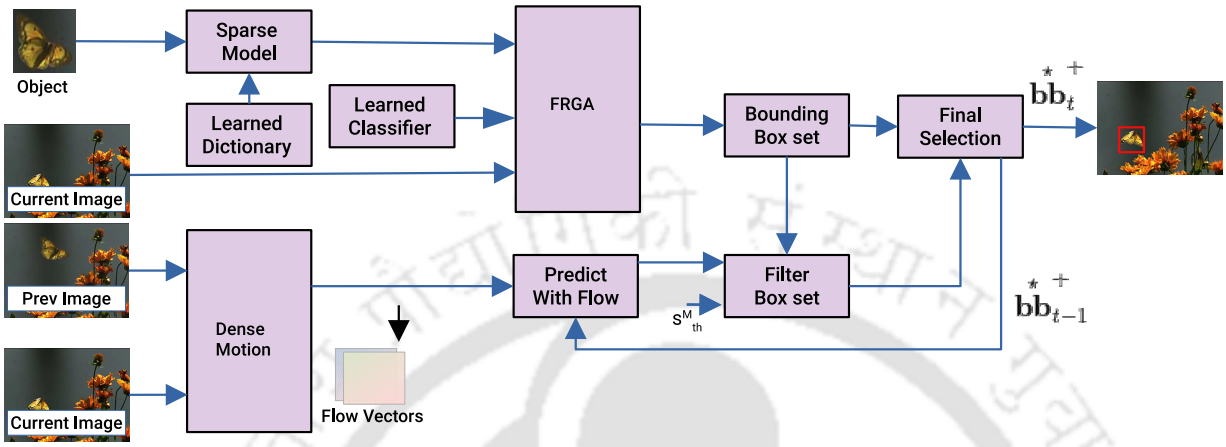

Figure 4.9: Functional block diagram of the combined approach with appearance and motion components.

4.7 Experiments and Results

The performance evaluation of the proposed trackers are reported on $VOT2018$, $OTB2015$ and $UAV123$ datasets. Measures used for evaluation include Accuracy, Robustness and Expected Average overlap for VOT dataset. The VOT toolkit is used for evaluation on $VOT2018$ dataset. The GOT-10K toolkit was used for evaluation on $OTB2015$ and $UAV123$ datasets. The success and precision plots are computed for these datasets.

4.7.1 Quantitative Analysis

The experiments were conducted to compare the two models as well as other trackers in the literature. Both object models were incorporated into a tracker that used FRGA for localization. Of the two models, it can be observed from Table 4.1 that the hybrid model (CWDSC) outperformed the generative model based tracker (DSC) in terms of accuracy, robustness and EAO. Accuracy and EAO are better with higher value (↑) and robustness is better when the value is lower (↓). The hybrid model can perform tracking for longer with lesser number of re-initalizations as compared to the generative model based tracker. With the addition of motion information, the performance improved for both the models highlighting its complementary nature. The hybrid model based tracker outperformed the generative model based tracker in both appearance-only and ensemble frameworks. For hybrid model, the improvement was observed in accuracy (from 0.332 to 0.351), robustness (from 6.477 to 6.396),

Algorithm 2 $DSC + FRGA + MOTION$

Input $\boldsymbol{\mathcal{I}}_0$, $\mathbf{bb}_0^+$
Output $\overset{\star}{\mathbf{bb}}{}_t^+$

Define $\boldsymbol{\mathcal{I}}^Q = extractROI(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_0^+)$
$\mathbf{D} = learnDict(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_0^{\pm})$
while $t \leq (n^T)$ **do** ▷ Loop through sequence
  $(\mathbf{U}_{(t-1)}, \mathbf{V}_{(t-1)}) = OpticalFlow(\boldsymbol{\mathcal{I}}_{t-1}, \boldsymbol{\mathcal{I}}_t)$
  $\overset{\star}{\mathbf{bb}}{}_{(t-1)}^+ = [x_{(t-1)}, y_{(t-1)}, w_{(t-1)}, h_{(t-1)}, \theta_{(t-1)}]$
  $\boldsymbol{\mathcal{I}}_t^S = extractROI(\boldsymbol{\mathcal{I}}_t, \overset{\star}{\mathbf{bb}}{}_{t-1}^+)$
  $\overset{\star}{\mathcal{P}}{}_t^M += \{(x,y)_{t-1} + (\mathbf{U}_{t-1}[x,y], \mathbf{V}_{t-1}[x,y]), (x,y)_{t-1} \in \overset{\star}{\mathcal{P}}{}_{(t-1)}^+\}$
  $\mathbb{P}_0$ = initalPopulation($\boldsymbol{\mathcal{I}}_t$, $\overset{\star}{\mathbf{bb}}{}_{t-1}^+$)
  while $\tau \leq n_\tau$ **do** ▷ FRGA iteration
    $\mathbb{P}^C_{t(\tau+1)}$ = childrenPopulation($\mathbb{P}_{(\tau)}$)
    while $i =\leq n_{\mathbb{P}}$ **do** ▷ Iterating through population
      $\mathbf{m}^{\mathbb{C}+}_{ti(\tau+1)}, \mathbf{m}^{\mathbb{C}-}_{ti\tau+1}$ = sparseModel($\boldsymbol{\mathcal{I}}_t$, $\mathbb{P}^C_{ti\tau+1}$)
      $s_{ti(\tau)} = \rho^{\pm}_{ti\tau}$
    end while
    $\mathbb{P}_{t(\tau+1)}$ = newPopulation($\mathbb{P}_{t(\tau)}, \mathbb{P}^C_{t(\tau)}, \mathbb{S}_{t(\tau)}$)
  end while
  while $i =\leq n_{\mathbb{P}}$ **do** ▷ Iterating through population
    $s^M_{ti} = \frac{|\mathcal{P}^M_t \cap \mathcal{P}_{tin_\tau}|}{|\mathcal{P}^M_t \cup \mathcal{P}_{tin_\tau}|}$
  end while
  Generate the set $\mathbf{J}_t$
  if $\mathbf{J}_t \neq \emptyset$ **then**
    $\overset{\star}{\mathbf{bb}}{}_t^+ := \mathbf{bb}^+_{ti^\star}$; $i^\star = \underset{i \in \mathbf{J}_t}{\operatorname{argmax}}\, \mathbb{S}^M_t$
  else
    $\overset{\star}{\mathbf{bb}}{}_t^+ := \mathbf{bb}^+_{ti^\star}$; $i^\star = \underset{i \in [1, n_{\mathbb{P}}]}{\operatorname{argmax}}\, \mathbb{S}_t$
  end if
end while

Algorithm 3 $CWDSC + FRGA + MOTION$

Input $\boldsymbol{\mathcal{I}}_0$, $\mathbf{bb}_0^+$

Output $\overset{\star}{\mathbf{bb}}{}_t^+$

Define $\boldsymbol{\mathcal{I}}^Q = extractROI(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_0^+)$
$\mathbf{D} = learnDict(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_0^{\pm})$
$\mathbb{BC} = box2cell(\mathbf{bb}_0^+)$
$\mathbb{M}^{\mathbb{T}}$ = sparseModel($\boldsymbol{\mathcal{I}}_0$,$\mathbb{BC}$, $\mathbf{D}$)
while $t \leq (n^T)$ **do** ▷ Loop through sequence
  $(\mathbf{U}_{(t-1)}, \mathbf{V}_{(t-1)}) = OpticalFlow(\boldsymbol{\mathcal{I}}_{t-1}, \boldsymbol{\mathcal{I}}_t)$
  $\overset{\star}{\mathbf{bb}}{}_{(t-1)}^+ = [x_{(t-1)}, y_{(t-1)}, w_{(t-1)}, h_{(t-1)}, \theta_{(t-1)}]$
  $\boldsymbol{\mathcal{I}}_t^S = extractROI(\boldsymbol{\mathcal{I}}_t, \overset{\star}{\mathbf{bb}}{}_{t-1}^+)$
  $\overset{\star}{\mathcal{P}}{}_t^M = \{(x,y)_{t-1} + (\mathbf{U}_{t-1}[x,y], \mathbf{V}_{t-1}[x,y]), (x,y)_{t-1} \in \overset{\star}{\mathcal{P}}{}_{(t-1)}^+\}$
  $\mathbb{P}_0$ = initalPopulation($\boldsymbol{\mathcal{I}}_t$, $\overset{\star}{\mathbf{bb}}{}_{t-1}^+$)
  while $\tau \leq n_\tau$ **do** ▷ FRGA iteration
    $\mathbb{P}^C_{t(\tau+1)}$ = childrenPopulation($\mathbb{P}_{(\tau)}$)
    while $i =\leq n_{\mathbb{P}}$ **do** ▷ Iterating through population
      while $r \leq n_r$ **do** ▷ Iterating through parts
        $\mathbb{M}^{\mathbb{C}}_{ti\tau}$ = sparseModel($\boldsymbol{\mathcal{I}}_t$, $\mathbb{P}^C_{ti\tau+1}$,$\mathbf{D}$,$\mathbb{BC}$)
        $s_{ti(\tau+1)} = \frac{1}{n_r}\sum_{r=1}^{n_r} \rho^+_{tir\tau}$
      end while
    end while
    $\mathbb{P}_{t(\tau+1)}$ = newPopulation($\mathbb{P}_{t(\tau)}, \mathbb{P}^C_{t(\tau)}, \mathbb{S}_{t(\tau)}$)
  end while
  while $i =\leq n_{\mathbb{P}}$ **do** ▷ Iterating through population
    $s^M_{ti} = \frac{|\mathcal{P}^M_t \cap \mathcal{P}_{tin_\tau}|}{|\mathcal{P}^M_t \cup \mathcal{P}_{tin_\tau}|}$
  end while
  Generate the set $\mathbf{J}_t$
  if $\mathbf{J}_t \neq \emptyset$ **then**
    $\overset{\star}{\mathbf{bb}}{}_t^+ := \mathbf{bb}^+_{ti^\star}$; $i^\star = \underset{i \in \mathbf{J}_t}{\operatorname{argmax}}\, \mathbb{S}^M_t$
  else
    $\overset{\star}{\mathbf{bb}}{}_t^+ := \mathbf{bb}^+_{ti^\star}$; $i^\star = \underset{i \in [1, n_{\mathbb{P}}]}{\operatorname{argmax}}\, \mathbb{S}_t$
  end if
end while

and EAO (from 0.055 *to* 0.057). The comparison of the hybrid (CWDSC+FRGA+MOTION) approach with other benchmarked trackers (classical approaches only) from the $VOT2018$ toolkit are presented in Table 4.2. The tracker performance was compared with 9 trackers available in $VOT2018$ challenge namely L1APG [9], Matflow [127], BDF [127], BST [127], DSST [62], FoT [127], IVT [54], MIL [36]. The proposed trackers provided competitive performance as compared to the other classical approaches. The evaluation of the proposed $CWDSC + FRGA + MOTION$ algorithm on $OTB2015$ dataset with precision and success plot are shown in Figures 4.10 4.11 respectively. The evaluation of the same on the $UAV123$ dataset are shown in Figures 4.12 and 4.13. For comparison with other trackers on $OTB2015$ dataset, the plots are shown in Appendix A.2. The use of motion information resulted in performance improvement for the $DSC + FRGA + MOTION$ tracker in accuracy (from 0.332 *to* 0.349), robustness (from 6.477 *to* 5.814), and EAO (from 0.55 *to* 0.62). A similar trend is observed with $CWDSC + FRGA + MOTION$. The performance improvement was observed in accuracy (from 0.351 *to* 0.359), robustness (from 6.396 *to* 5.640) and EAO (from0.57 *to* 0.66).

Name	**Accuracy** (↑)	**Robustness** (↓)	**EAO** (↑)
DSC+FRGA	0.332	6.477	0.055
CWDSC+FRGA	0.351	6.396	0.057
DSC+FRGA+MOTION	0.349	5.814	0.062
CWDSC+FRGA+MOTION	0.359	5.640	0.066

Table 4.1: Performance comparison of hybrid and generative approaches on $VOT2018$ dataset with Accuracy (A), Robustness (R) and Expected Average Overlap is shown. Performance improvement with the addition of motion information is shown in the table as well. Evaluation is done on 60 sequences available in the dataset.

The overall computational complexity of the tracker depends on the population size ($n_{\mathbb{P}}$), dictionary size ($n_{\mathfrak{p}} \times n_a$), number of OMP iterations (n_τ), number of candidate region patches($n_{\mathbb{X}+}$) and the computations required for evaluating the orthogonal projection for OMP.

4.7.2 Qualitative Analysis

The tracking output on sequences from $VOT2018$ dataset is shown in Figure 4.14. The performance is similar to some of the classical approaches reported in the literature. However, the sparse representation based tracker provided lower performance compared to color based tracker on $OTB2015$ and $UAV123$ datasets. A patch based framework makes the object model less flexible in comparison to a global feature model like color distribution. Here, the later is relatively robust to in-plane rotation,

Name	Accuracy(↑)	Robustness(↓)	EAO(↑)
L1APG	0.410	8.066	0.07
Matflow	0.386	4.973	0.091
BDF	0.355	4.476	0.092
BST	0.282	4.086	0.118
DSST	0.388	5.430	0.079
FoT	0.388	4.190	0.129
(CWDSC+FRGA+MOTION)	0.359	5.640	0.066
IVT	0.382	6.093	0.076
MIL	0.383	3.933	0.117
Struck2011	0.410	4.739	0.095

Table 4.2: Performance comparison on $VOT2018$ dataset with the baseline scheme in $VOT2018$ toolkit. Evaluation is done on 60 sequences available in the dataset.

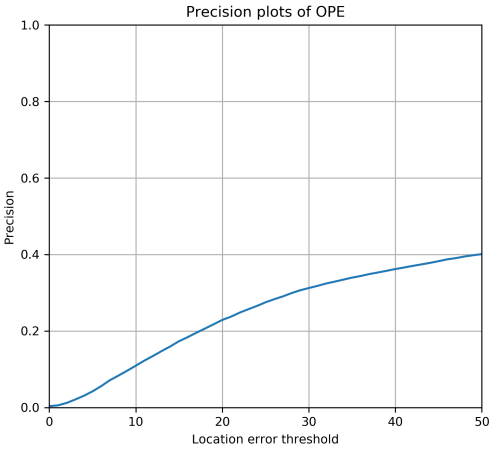

Figure 4.10: Precision plot of proposed tracker on $OTB2015$ dataset.

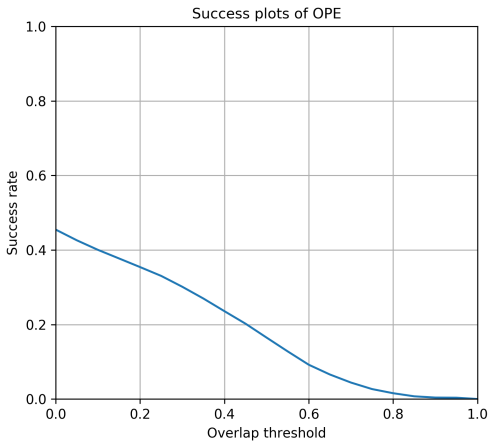

Figure 4.11: Success plot of proposed tracker on $OTB2015$ dataset.

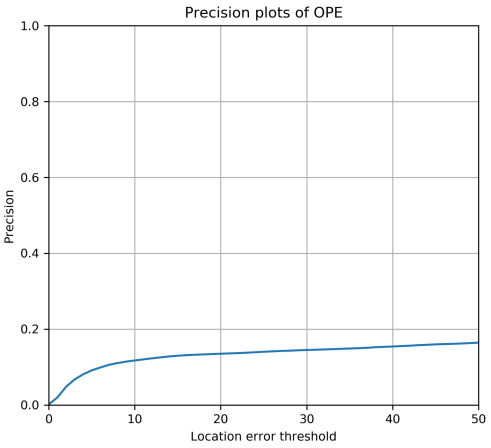

Figure 4.12: Precision plot of proposed tracker on $UAV123$ dataset.

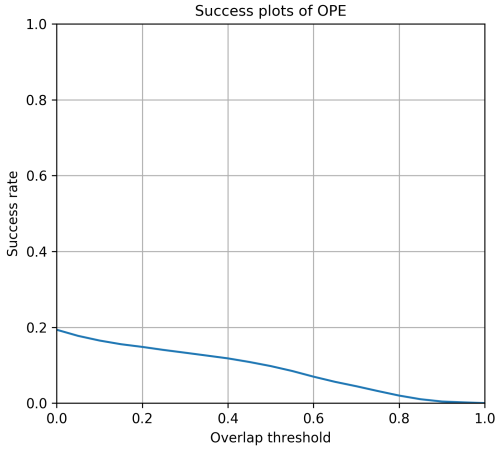

Figure 4.13: Success plot of proposed tracker on $UAV123$ dataset.

scaling and other appearance changes. However, such changes will impact the pixel relationships in a patch based model. Some flexibility in object model is brought in with the multi-part approach. However, the object model needs to be updated to accommodate the appearance changes.

Such an update is not explored due to the time complexity involved in dictionary learning in the current formulation resulting in the degraded performance. Most sequence have a complex background and the background information also changes in the sequences. Here, the background is explicitly modeled and the same needs to be updated to capture the changed context as well. Trained binary classifier discriminates against the background and helps in handling the updated context. This gives it a slightly better performance compared to the generative model. The VOT evaluation has a reinitialization procedure upon track loss. Also, the sequences are of shorter length when compared to other datasets. The $OTB2015$ and $UAV123$ datasets have relatively longer sequences. Also, the GOT-10k toolkit does not perform reinitializations. Both these properties make the performance values low in these datasets for the proposed tracker.

4.8 Discussions

This chapter has attempted to capture the local structural information of object appearance in the sparse representation framework. Two models are proposed in this context. The first one in the generative framework and the second one in the hybrid framework. Both proposals model the object appearance using the distribution of sparse codes of vectorized object patches. In the first model proposal, dictionaries are learned on both foreground and background to bring discrimination. This is enhanced further in the second proposal by learning a binary classifier that assigns weights to the sparse codes. Novel objective functions as similarity between target and candidate models were proposed on the developed models. The object localization was performed with the *FRGA* algorithm. Further, complementary information from inter-frame dense motion was used to enhance the tracker performance.

The present work did not incorporate continuous dictionary and classifier update schemes in the object model as well as the background. This extension would have enabled the tracker to trail targets for longer duration, possibly under appearance changes and occlusions. Also, the use of K-SVD and ℓ_1 minimization makes the tracker slower. The present formulation can be made to run faster by incorporating block K-SVD and block OMP based approaches.

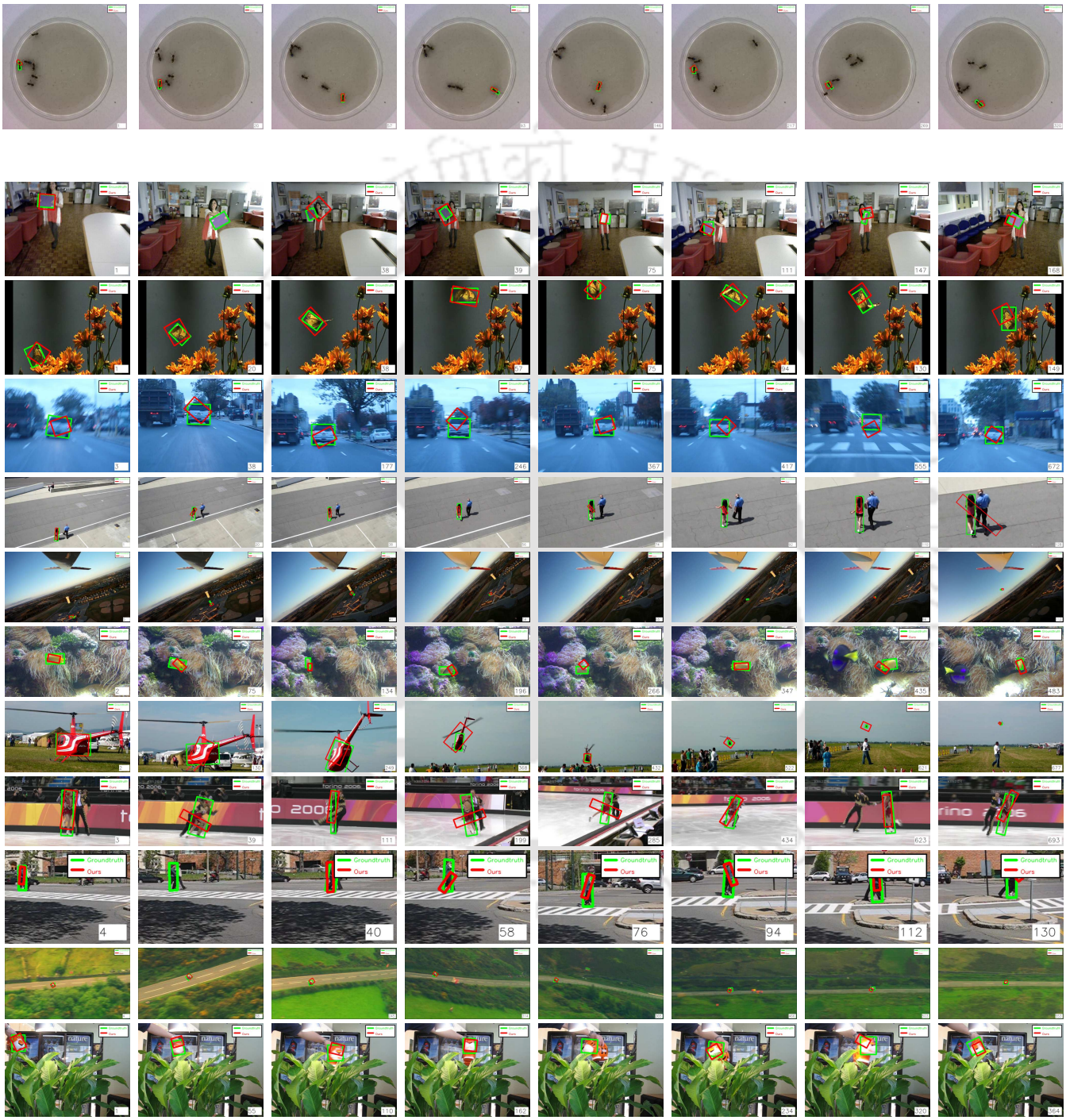

Figure 4.14: Each row shows the respective outputs of visual tracking on sequences from $VOT2018$ dataset. Groundtruth is marked in green and tracker prediction is colored red.

This work used the structural information of the object appearance in sparse representation framework. However, the model learning was mostly restricted to the data available from the first frame of the sequence. Recent tracker formulated in deep learning framework use networks capable of producing rich representations from object appearance. Such networks are trained on larger datasets and are shown to be much more generalizable [283]. Accordingly, this thesis proposes a tracker in the deep learning framework and is presented in the next chapter.

5

Siamese Network based Tracking

Publications

- **Mathew Francis**, Prithwijit Guha, "Siamese Fully Convolutional Tracker with Motion Correction", *International Conference on Pattern Recognition (ICPR), 2021*.

Contents

Overview

The siamese networks are trained to predict the similarity between its inputs. Object tracking can be formulated in this framework. Many successful trackers are based on the siamese architecture due to their performance and efficiency. This formulation computes the similarity between the appearance feature embedding of the target object and the candidate regions within a search image region. This leads to a scoremap of similarity values. Trackers are often challenged by occlusion of target object. To this end, this chapter proposes a multi-part object appearance model (in siamese framework) to handle occlusions. However, only appearance embedding based tracking might not be sufficient to handle similar looking objects (distractors) or background clutter. Motion information is complementary to appearance and is known to improve tracking performance. Thus, the proposed tracker further use motion information to perform a re-ranking of appearance based predictions. The part-based object tracker with motion based re-ranking is evaluated on $VOT2018$, $VOT2019$, $OTB0215$ and $UAV123$ datasets.

5.1 Siamese Trackers: An Introduction

A siamese network (SN) has two parallel branches hosting two sub-networks receiving two inputs. These two sub-networks compute the embeddings of the two inputs. A correlation between these embeddings is produced at the output of the SN [128]. The SN is trained to predict the similarity of its two inputs. Here, a pseudo-siamese architecture is followed as the weights are shared between the two branches [131]. That is, both branches host the same sub-networks with same weights for embedding computation. Object tracking can be formulated as a problem of matching target and candidate with similarity learning through a siamese network [131]. However, the matching should be able to handle the variations that an object can undergo in a sequence. The *Siamese Network based Trackers* (SNTs) belong to the class of matching based trackers where feature embeddings are compared [101]. The SNTs are considered as one of the most promising designs in tracking on account of their ability to balance performance and efficiency [132–134]. Many of the recent SNTs are also fully convolutional (FC) [10] in nature, thereby allowing them to work with inputs of variable sizes. Among other things, this allows for an end-to-end training network to be designed with fewer hyper-parameters [135].

Two images are provided as input to a SNT. First, an image containing the target object. Second, the search image in which the object is to be located. The SNTs operate in two stages. The first

stage extracts feature embeddings from the input images using CNN based sub-networks. The second stage computes the correlation between these embeddings. Bertinetto *et al.* [10] proposed a fully convolutional siamese (SiamFC) tracker with cross-correlation operation performed on the embeddings. Initial SNTs used the AlexNet [124] architecture as the CNN sub-network. The weights of this sub-network were fine-tuned for tracking on new training data [10, 137, 138]. The performance of these SNTs were limited by the representation generation capability of the relatively shallow AlexNet [139]. Deeper networks can produce richer semantic features. Thus, recent SNTs use deeper and wider sub-networks like ResNet [140], VGG-19 [14], Inception [141] among others. Li *et al.* [139] improved the SNT performance with a custom *VGG-16* [14] sub-network. In this chapter, the SiamFC [10] tracker with a *VGG-16* CNN based sub-network is used as baseline.

Tracker drift or track loss are often caused by object occlusions [153]. The central pixels or object parts have lesser chances of getting occluded as compared to the peripheral ones. This assumption is exploited by most existing works. Comaniciu *et al.* [4] used Epanechnikov kernel weighted color distribution that assigned higher weights to central pixels as compared to ones at the object boundary. Wu *et al.* [154] used anti-occlusion region proposal network (RPN) to address this issue. Existing works have mostly addressed the issue of partial occlusion by separately tracking object parts. Adam *et al.* [102] tracked different parts of the object and the scoremaps from individual parts were merged by weighted averaging of individual score maps of parts for obtaining the final prediction.

The SNTs use rich feature embeddings generated by the deep CNN based sub-networks. They extract the semantic information from the input images which are used in correlating the target and the candidates. However, presence of similar objects a.k.a distractors pose a problem to SN. Distractors can be detected with multiple peaks in the tracker response map. Zhu *et al.* [150] addressed this by performing hard negative mining during training. Li *et al.* [151] proposed a re-detection mechanism to handle distractors based on [137]. The re-detection approach uses a classifier to re-rank potential candidates [151] and follow a tracking-by-detection approach. This was pushed further with a two stage detector model proposed by Voigtlaender *et al.* [152]. The SNTs are developed on appearance-only and hence lose any temporal information between frames. The temporal information provides additional cues to handle distractors in tracking. Distractors have same appearance as the target object but, they might not be having the same motion signature as that of the object. In this work, motion cues are used for re-ranking the predictions from the appearance component (SiamFC) to improve the

tracking performance.

Classical computer vision techniques like Horn–Schunck [91] or Farneback [91] optical flow can be used to estimate inter-frame dense motion. Recent works have formulated the motion estimation problem in the deep learning framework [284–286]. This work uses the *LiteFlowNet* [18] which is a relatively lighter network with good performance. Motion and appearance are considered complementary information in tracking, and their combination in an ensemble is expected to enhance tracking performance [77].

This chapter proposes an ensemble of (a) multi-part object appearance based siamese tracker and (b) motion based refinement of appearance tracker predictions. The *SiamFC* model with a pre-trained *VGG-16* based sub-network forms the appearance component. The object region is divided into five parts, v.i.z. top-left, top-right, bottom-left, bottom-right and central. These five parts are independently tracked. The scoremaps obtained from the five parts are combined through weighted average (central part having highest weight). The combined scoremap obtained from the appearance component is used in selecting potential candidate locations of the object. Inter-frame dense motion is estimated by using the *LiteFlowNet*. The motion information is used to re-rank the appearance based predictions. The major contributions of this work are the following.

- Multi-part object model for handling occlusions
- Motion information based re-ranking for refining the appearance component prediction

The proposed tracker is evaluated with the VOT toolkit on $VOT2018$ dataset and GOT-10k toolkit on $VOT2019$, $OTB2015$ and $UAV123$ datasets. The proposed tracker outperforms the baseline SiamFC tracker and provides competitive performance with respect to others.

Rest of the chapter is organized as follows. Section 5.2 briefly discusses the existing works that are relevant to the proposed approach. Section 5.3 describes the proposed single-part (whole object) based appearance tracker. Section 5.4 explains the proposed multi-part object model based appearance tracker. Section 5.5 describes the motion information based re-ranking approach. Section 5.6 presents the experimental results and compares them with the baseline Siamese trackers and other benchmark trackers in literature. Section 5.7 summarizes the current work and discusses possible future extensions.

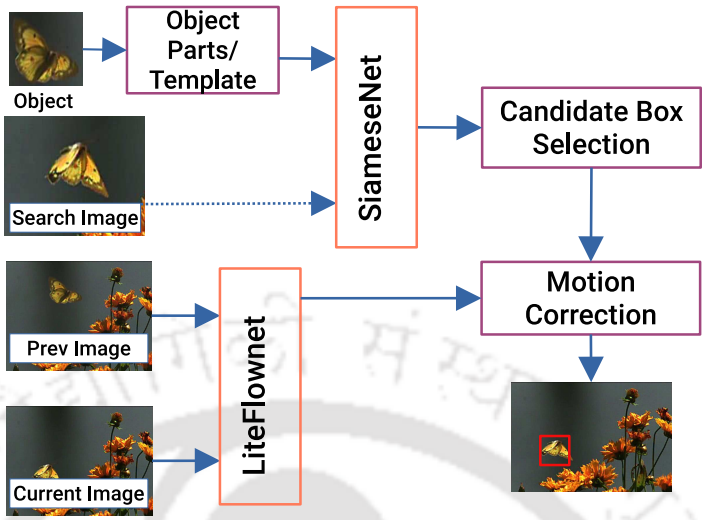

Figure 5.1: The functional block diagram of the approach showing the appearance and motion components.

5.2 Related Work

This work explores siamese network based tracker with CNN based motion estimation for object tracking. Accordingly, the siamese network based tracking formulation subsection 5.2.1 and CNN based inter-frame motion estimation(subsection 5.2.2) are briefly reviewed next. Embeddings from a pre-trained siamese network is used for modeling. For localization, a similarity search with correlation operation is performed. This forms the appearance component of the algorithm. Additional cues in the form of motion information estimated from a deep neural network is explored in subsection 5.2.2. Siamese based tracking approach is described in detail next.

5.2.1 Siamese Fully Convolutional Tracker

Bertinetto *et al.* [10] formulated tracking using siamese network architecture. Their proposal, *SiamFC* accepts two images as inputs. First the search image $\mathcal{I}_t^S \in \mathbb{R}^{255\times255\times3}$ and second the query image $\mathcal{I}^Q \in \mathbb{R}^{127\times27\times3}$. $\mathcal{I}^Q$ is derived from the bounding box annotation $\mathbf{bb}_0^+$ of initial frame $\mathcal{I}_0$. It is otherwise called as the template image. Search image $\mathcal{I}_t^S$ is a larger image (compared to $\mathcal{I}^Q$) on which the target object is to be localized. The search region $\mathcal{I}_t^S$ is defined in $\mathcal{I}_t$ as a region concentric with $\overset{\star}{\mathbf{bb}}_{t-1}^+$, but with twice the width and height of $\overset{\star}{\mathbf{bb}}_{t-1}^+$.

Siamese network has sub-network ϕ_{sub} with shared weights. It uses the *VGG-16* backbone. Siamese network $\mathbf{\Phi}$ is trained on positive $(\mathbb{X}_p^+)$ and negative $(\mathbb{X}_p^-)$ pairs extracted from pair of frames that are

t_0 frames apart. A positive pair consists of an anchor image and a similar image while a negative pair consists of anchor and a dissimilar image. Object class information is discarded during dataset preparation. Image regions $\mathcal{I}_t^Q$ and $\mathcal{I}_{t+t_0}^S$ are respectively extracted from $\mathcal{I}_t$ and $\mathcal{I}_{t+t_0}$ the pair of images. Here, $\mathcal{I}_t^Q$ is the anchor image and the similar and dissimilar images are extracted from $\mathcal{I}_{t+t_0}^S$. Both images $\mathcal{I}_t^Q$ and $\mathcal{I}_{t+t_0}^S$ are centred based on the ground-truth annotations $a(\mathcal{I}_t)$ and $a(\mathcal{I}_{t+t_0})$ respectively. The network is trained on logistic loss [287] (equation 5.1).

$$\mathfrak{L}(y, \hat{y}) = log(1 + \exp(-y\hat{y})) \tag{5.1}$$

Here, $y \in \{+1, -1\}$ refers to the ground-truth label and $\hat{y}$ is the predicted one. However, since the query image $\mathcal{I}_t^Q$ is small in size compared to $\mathcal{I}_{t+t_0}^S$, multiple candidate images $\mathcal{I}_{t+t_0}^{\mathfrak{C}}$ can be extracted from it to form the pairs for training. Hence, the output of the network is a scoremap instead of a single score. Let $\mathcal{P}_{t+t_0}^S$ be the set of all pixel positions in the scoremap $\mathbb{S}_{t+t_0}$. The loss is defined on the scoremap as follows

$$\mathcal{L}(Y, \hat{Y}_{t+\tau}) = \frac{1}{|\mathcal{P}_{t+t_0}^S|} \sum_{\mathbf{p} \in \mathcal{P}_{t+t_0}^S} \mathfrak{L}(Y([\mathbf{p}], \hat{Y}[\mathbf{p}]) \tag{5.2}$$

where, $Y[\mathbf{p}] \in \{+1, -1\}$ is the ground-truth label for position $\mathbf{p}$ in scoremap $\mathbb{S}_{t+t_0}$, $Y, \hat{Y}$ are arrays with the ground-truth and predicted values respectively of all pixel positions in $\mathcal{P}_{t+t_0}^S$. Positive pairs are assigned a label $Y[\mathbf{p}] = +1$ and negative pairs are assigned label $Y[\mathbf{p}] = -1$ based on the following rule.

$$Y[\mathbf{p}] = \begin{cases} +1, & \text{If } \lambda_s \|\mathbf{p} - \mathbf{c}\| \leq \lambda_r \\ -1, & \text{Otherwise} \end{cases}$$

where λ_r is a radius value, $\mathbf{c}$ is the centroid of output scoremap of the network and λ_s is the stride of the network. The above rule for defining ground-truth label results in fewer positive pairs as compared to negative pairs. Thus, weighted loss is used to address this issue of class imbalance. The use of motion information in tracking is discussed next.

5.2.2 Flow Networks

Optical flow is defined as the pixel motion between adjacent frames [91]. Traditional approaches use variational techniques that are solved as an energy minimization problem. Recent approaches follow optical flow estimation with CNNs in a data driven learning paradigm. Horn and Schunck [89]

proposed the first approach to estimate the optical flow. Lucas and Kanade [288] proposed local constraint based approach to estimate sparse optical flow. Large displacement between adjacent frames was a common problem in optical flow estimation. Brox *et al.* [289] proposed a pyramidal approach to overcome this. Similar approach is seen in [290, 291]. Forward-backward consistency check is employed in many approaches [292–294] to overcome the effects of occlusion. Several approaches [289,295] are proposed counter illumination variations that violate the brightness constancy assumption of optical flow computation. Recent approaches [284–286,296–298] use CNNs to estimate optical flow. These methods learn to compute optical flow directly from data.

Dosovitskiy *et al.* [284] proposed the first CNN based approach (*FlowNet*) for optical flow estimation. Ilg *et al.* [296] proposed an improved version *FlowNet2.0* by stacking the sub-networks. Xiang *et al.* [299] introduced brightness constancy as a prior to improve the prediction. Ranjan *et al.* [300] predicted multi-resolution optical flow with the help of a pyramidal network. Liu *et al.* [297] presented a self-supervised learning approach that used temporal information to improve the estimation. Sun*et al.* [286] proposed an modification to the spatial pyramid network where feature warping was used instead of image warping. Teed *et al.* [301] proposed a transformer with a recurrent update operator. More on CNN based optical flow algorithms can be found in [302, 303].

Hui *et al.* [18] have proposed a lighter network architecture (*LiteFlowNet*) for inter-frame motion estimation. The *LiteFlowNet* [18] has one of the lighter architectures in the domain. Compared to *FlowNet2.0* the model is 30 times smaller at the same time it is faster by a factor of 1.36 [18]. It uses the pyramidal feature extraction similar to other networks [300]. However, feature warping [286] is used rather than image warping [296] to estimate flow. It uses a descriptor matching network to compare the embeddings of the image pairs. It relies on a cascaded flow inference based on cost volume [286] and sub-pixel [18] refinement. Local convolution layer is included to provide flow regularization. The functional block diagram of this approach is shown in Figure 5.2. This work uses the *LiteFlowNet* [18] for inter-frame dense motion estimation.

5.3 Single Part Model

Consider the target object template image $\mathcal{I}^Q \in \mathbb{R}^{h_q \times w_q \times 3}$ constructed from initial frame annotation region $\mathbf{bb}_0^+$ in first frame $\mathcal{I}_0$. Let $\mathcal{I}_t^S$ be the search image in the t^{th} frame $\mathcal{I}_t$. A correlation operation is performed between the embeddings of the inputs to the siamese network to get a score

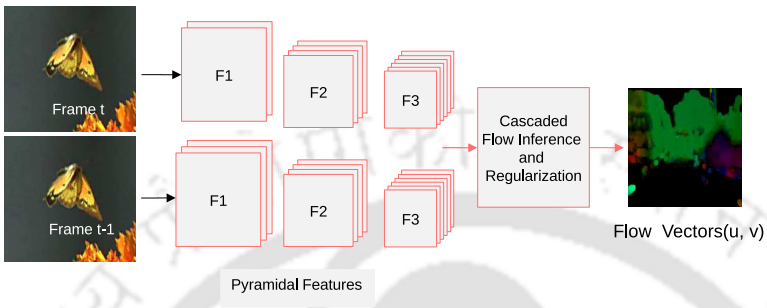

Figure 5.2: Block diagram for *LiteFlowNet* flow prediction network

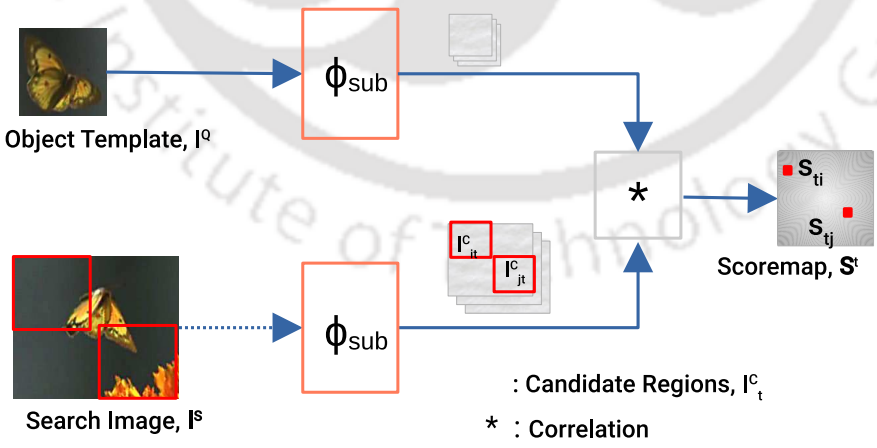

Figure 5.3: Block diagram of baseline Siamese tracker

map $\mathbb{S}$. The correlation computation through a siamese architecture can be represented in the following manner.

Let $\mathbb{B}_t^+ = \{\mathbf{bb}_{ti}^+; i = 1, \ldots, n_{\mathbb{B}}\}$ be the set of candidate region proposals that are constructed within $\boldsymbol{\mathcal{I}}_t^S$. Let $\boldsymbol{\mathcal{I}}_{ti}^{\complement}$ be the candidate template image constructed by using the candidate region proposal $\mathbf{bb}_{ti}^+$ and $\boldsymbol{\mathcal{I}}_t$. The siamese architecture contains a deep sub-network $\phi_{sub}(\bullet)$ that operates on both $\boldsymbol{\mathcal{I}}^Q$ and $\boldsymbol{\mathcal{I}}_{ti}^{\complement}$ and works as the feature extractor network. Let E^Q and $E_{ti}^{\mathbb{C}}$ be the respective feature embeddings extracted by ϕ_{sub} from inputs $\boldsymbol{\mathcal{I}}^Q$ and $\boldsymbol{\mathcal{I}}_{ti}^{\complement}$

$$E^Q = \phi_{sub}(\boldsymbol{\mathcal{I}}^Q) \tag{5.3}$$

$$E_{ti}^{\mathbb{C}} = \phi_{sub}(\boldsymbol{\mathcal{I}}_{ti}^{\complement}) \tag{5.4}$$

A correlation operation (denoted by $\star$) is performed between the embeddings to estimate the similarity of $\boldsymbol{\mathcal{I}}^Q$ and $\boldsymbol{\mathcal{I}}_{ti}^{\complement}$ and is given by

$$\rho_{ti}^+ = E^Q \star E_{ti}^{\mathbb{C}} \tag{5.5}$$

Note that the target embedding E^Q is pre-computed. Thus, only $n_{\mathbb{B}}$ number of correlation estimation computations are performed in each t^{th} frame. Optimal candidate region proposal $\overset{\star}{\mathbf{bb}}_t^+$ is chosen as the one with highest similarity with target template mage.

$$\overset{\star}{\mathbf{bb}}_t^+ = \mathbf{bb}_{ti^\star}^+ \;\; s.t. \;\; i^\star = \underset{i=1,\ldots,n_{\mathbb{B}}}{\arg\max}(E^Q \star E_{ti}^{\mathbb{C}}) \tag{5.6}$$

Object model based on a single template image is widely used in tracking literature. Multi-part object models have been widely used in tracking for occlusion handling [15, 102, 282]. This work explores a similar work and is explored next.

5.4 Multi-Part Object Model

Occlusion is one of the common challenges in tracking. An object may get occluded either partially or completely by other scene objects [304]. Various approaches are proposed in the literature to address the issue of occlusion. One such approach is the part-based one. It is based on the observation that when an object gets occluded, the action is gradual. During partial occlusion , a few parts of the object are still visible and can still be used for tracking. This makes it easier to track the visible parts

rather than the entire target. In this section, a part-based model is explored. This approach uses deep network embedding of each object part.

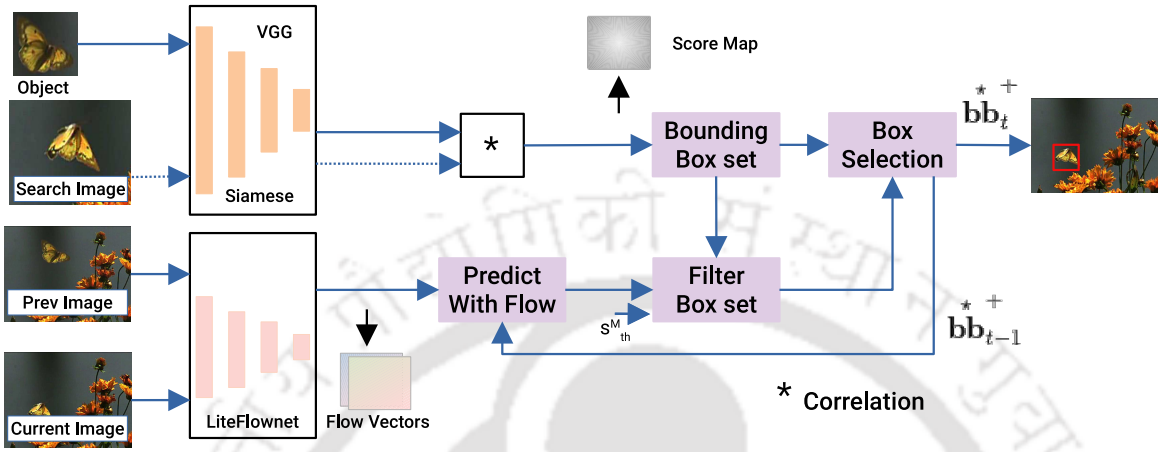

Figure 5.4: Functional block diagram of the single template based approach. Candidate predictions from the template based appearance component is re-ranked with the motion information.

The object appearance is modeled as the embedding $E^Q = \Phi_{sub}(\mathcal{I}^Q)$ of the template image $\mathcal{I}^Q$. Embeddings are extracted from $\text{Conv}5-3$ layer as shown in (Figure 5.5) of $VGG{-}16$ [305]. The *SiamFC* computes the embeddings of the entire object template image $\mathcal{I}^Q$. In contrast, multi-part proposal splits $\mathcal{I}^Q$ into different predefined parts and use their individual embeddings for similarity score map generation. The object template image $\mathcal{I}^Q$ is divided into $n_p = 5$ different parts. Let the set of *object template parts* corresponding to top-left ($\mathcal{I}^Q_1$), top-right ($\mathcal{I}^Q_2$), bottom-left ($\mathcal{I}^Q_3$), bottom-right ($\mathcal{I}^Q_4$) and central ($\mathcal{I}^Q_5$) be denoted by $\mathbb{Q} = \{\mathcal{I}^Q_1, \mathcal{I}^Q_2, \mathcal{I}^Q_3, \mathcal{I}^Q_4, \mathcal{I}^Q_5\}$, where $\mathcal{I}^Q_r, \in \mathbb{R}^{\frac{w_0}{2} \times \frac{h_0}{2}}, r = 1, \ldots, n_r$. Search image $\mathcal{I}^S_t \in \mathbb{R}^{w_s \times h_s \times 3}$. The object template and its parts are illustrated in Figure 5.6.

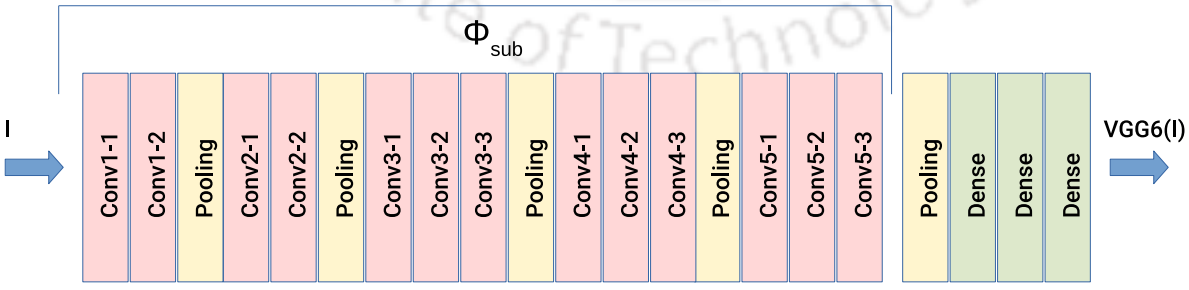

Figure 5.5: Architecture diagram of $VGG{-}16$ network

The siamese network computes the score-map of each object template part of the query image $\mathcal{I}^Q$ on the search image $\mathcal{I}^S_t$. Let $\{\mathbb{S}_{1t}, \mathbb{S}_{2t}, \mathbb{S}_{3t}, \mathbb{S}_{4t}, \mathbb{S}_{5t}\}$ be the set of score-maps corresponding to the respective object template parts in $\mathcal{I}^Q$. The siamese network $\mathbf{\Phi}$ computes the similarity scores between

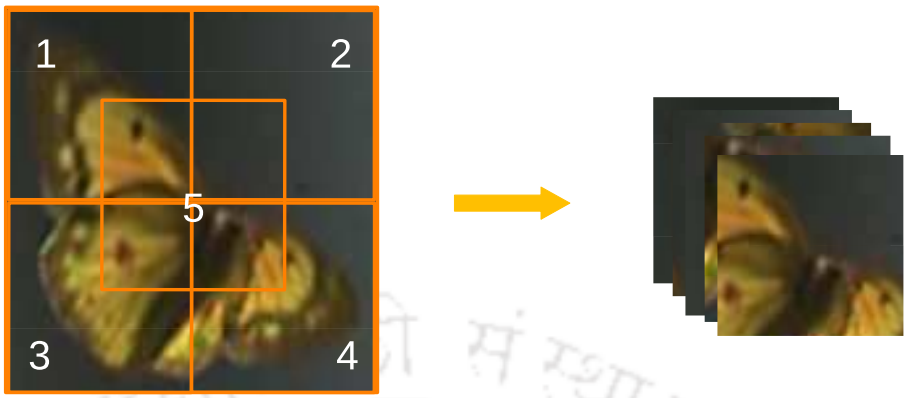

Figure 5.6: Template divided into multiple parts.

$\mathcal{I}^Q$ and candidate regions $\mathcal{I}^{\complement}_{ti} \in \mathcal{I}^Q_t$, $i = 1, \ldots, n_c$. Φ_{sub} compute the embeddings of both $\mathcal{I}^Q$ and $\mathcal{I}^{\complement}_{ti}$. Similarity is computed with correlation operation ($\star$) on each object part and candidate region. Score $\rho^+_{tir}, r = 1, \ldots, n_p$ (Equation 5.7) is computed as

$$\rho^+_{tir} = \mathbf{\Phi}\left(\mathcal{I}^Q_r, \mathcal{I}^{\complement}_{ti}\right) = \phi_{sub}(\mathcal{I}^Q) \star \phi_{sub}(\mathcal{I}^{\complement}_{tir}) \tag{5.7}$$

A candidate patch $\mathcal{I}^{\complement}_{ti}$ has scores coming from its individual parts $\mathcal{I}^Q_r$. Scoremaps at the output of the network is scaled to the size of $\mathcal{I}^S_t$ ((w_s, h_s)). Individual scoremaps are combined as follows.

$$\begin{aligned}\mathbb{S}_{ti}(x,y) = w_1 \times \mathbb{S}_{ti1}(x_{ti} - \frac{w_0}{2}, y_{ti} - \frac{h_0}{2}) + w_2 \times \mathbb{S}_{ti2}(x_{ti} + \frac{w_0}{2}, y_{ti} - \frac{h_0}{2}) + \\ w_3 \times \mathbb{S}_{ti3}(x_{ti} - \frac{w_0}{2}, y_{ti} + \frac{h_0}{2}) + w_4 \times \mathbb{S}_{ti4}(x_{ti} + \frac{w_0}{2}, y_{ti} + \frac{h_0}{2}) + w_5 \times \mathbb{S}_{ti5}((x_{ti}, y_{ti})\end{aligned} \tag{5.8}$$

where, $w_r, r = 1, \ldots, 5$ are the weights and $\sum_{r=1}^{5} w_r = 1$. Also, $w_r = \frac{1}{5}, r = 1, \ldots 4$ and $w_5 = \frac{2}{5}$. Center part of the object is given more weight as it is the part which has the least chance to get occluded compared to parts at the boundary. The combination provides with a single similarity map $\mathbb{S}_t = \{\rho_{ti}, i = 1, \ldots, n_c\}$ as output. A score map $\mathbb{S}_t = \{\rho^+_{ti}\}$ shows the likelihood of finding object in the region $\mathbf{bb}^+_{ti}$ centered at (x_{ti}, y_{ti}). Here, each candidate patch $\mathcal{I}^{\complement}_{ti}$ is characterised by its bounding box $\mathbf{bb}^+_{ti}$. With only the appearance component the bounding box of object is predicted with scoremap as

$$\overset{\star}{\mathbf{bb}}{}^+_t = \mathbf{bb}^+_{ti^\star}; i^\star = \underset{i=1,\ldots,n_c}{\operatorname{argmax}}(S_t) \tag{5.9}$$

where, $\overset{\star}{\mathbf{bb}}{}^+_t$ is the predicted object location from the siamese network. Additional information like motion cue is used to further enhance the tracking performance and is explained next.

5.5 Motion Correction

Motion is complementary to appearance and is widely explored in tracking literature [34, 77, 96]. Combining algorithms that use complementary information are shown to improve tracking performance [306]. Also, motion information often helps in discriminating target from distractors having similar appearance features. Let the dense motion components along horizontal (x) and vertical (y) directions be $\mathbf{U}_{t-1}$ and $\mathbf{V}_{t-1}$ respectively. The motion associated to target and distractors might be different. In this proposal, predictions from the appearance component are re-ranked with motion information to select the optimum bounding box $\overset{\star}{\mathbf{bb}}{}^{+}_{t}$.

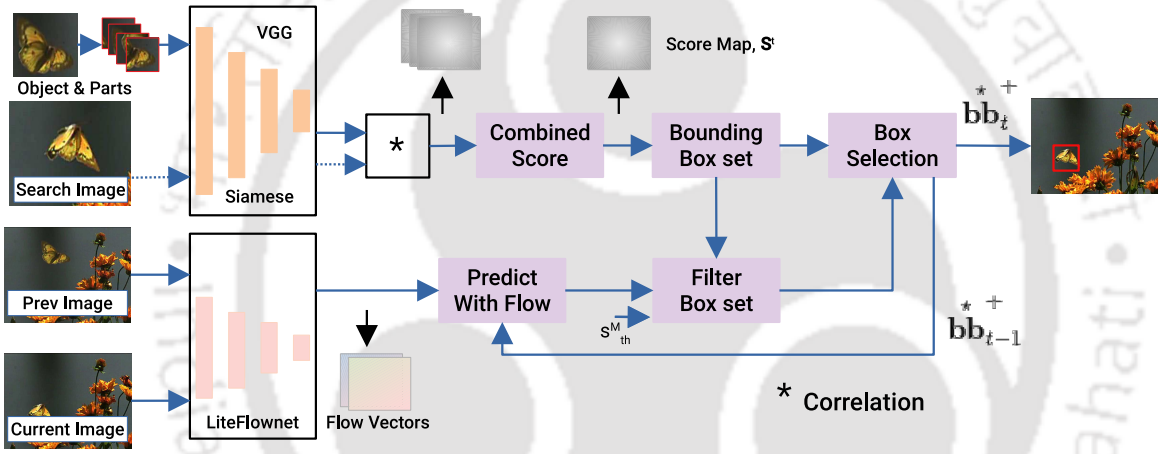

Figure 5.7: Functional block diagram of the multi-part template based approach. Candidate predictions from the template based appearance component is re-ranked with the motion information.

Inter-frame motion from *LiteFlowNet* [18] is used to refine predictions from appearance component to localize the target in current frame $\boldsymbol{\mathcal{I}}_t$. Per-pixel dense motion is estimated from two consecutive images $\boldsymbol{\mathcal{I}}_{t-1}$ and $\boldsymbol{\mathcal{I}}_t$. Let $\mathbf{bb}^{+}_{t-1}$ be the predicted bounding box from the last frame $\boldsymbol{\mathcal{I}}_{t-1}$. Let the dense motion vector matrices from *LiteFlowNet* [18] be $\mathbf{U}_{t-1}, \mathbf{V}_{t-1}$.

Let the set of all pixel coordinates belonging to bounding box $\overset{\star}{\mathbf{bb}}{}^{+}_{t-1}$ be defined as $\overset{\star}{\mathcal{P}}{}^{+}_{t-1}$. Let the set of all pixel positions within a candidate bounding box $\mathbf{bb}^{+}_{ti}$ be represented as $\overset{\star}{\mathcal{P}}{}^{+}_{ti} = \{(x_j, y_j)_{ti} \| x_{min} < x_j < x_{max}, y_{min} < y_j < y_{max}\}$. Here, (x_{min}, y_{min}) and (x_{max}, y_{max}) are the respective top-left and bottom-right pixel co-ordinates of $\overset{\star}{\mathbf{bb}}{}^{+}_{ti}$. These pixel-sets are used further to incorporate the motion correction. Next frame position of very pixel in $\overset{\star}{\mathcal{P}}{}^{+}_{(t-1)}$ can be predicted from motion to form the pixel set in $\mathcal{P}^{M}_{t}$ and is given by

$$\mathcal{P}^{M}_{t} = (x, y)^{M}_{t} = (x, y)_{t-1} + (\mathbf{U}_{t-1}[x, y], \mathbf{V}_{t-1}[x, y]);\ (x, y)_{(t-1)} \in \overset{\star}{\mathcal{P}}{}^{+}_{(t-1)} \tag{5.10}$$

where, the pixel intensity at $(x,y)_t \in \boldsymbol{\mathcal{I}}_{t-1}$ is predicted to reach position $(x,y)_t \in \boldsymbol{\mathcal{I}}_t$ using motion $\mathbf{U}_{t-1}[x,y], \mathbf{V}_{t-1}[x,y]$ is the motion vector components at position (x,y). Let $\mathcal{P}_t^M$ represent pixel position set $\mathcal{P}_t^M = \{(x,y)_t^M\}$.

The potential target bounding boxes are first predicted from the output score map of the appearance tracker with equation 5.9. Each position in $\boldsymbol{\mathcal{I}}_t^S$ is a potential candidate. The top-k values from the score map $\mathbb{S}_t$ correspond to the k best predicted bounding boxes from the appearance component. The set of these top-k bounding boxes ($\mathbb{B}_t^k$) and their corresponding scores ($\mathbb{S}_t^k$) are defined as follows.

$$\mathbb{B}_t^k = \{\mathbf{bb}_{ti}^+ : \rho_{ti}^+ \in top-k(\mathbb{S}_t)\} \tag{5.11}$$

$$\mathbb{S}_t^k = \{s_{ti} : \rho_{ti}^+ \in top-k(\mathbb{S}_t)\} \tag{5.12}$$

The motion predicted pixel-set $\mathcal{P}_t^M$ is used to select one from the k bounding boxes in $\mathbb{B}_t$. Let s_{ti}^M be the overlap between $\mathbf{bb}_{ti}^+ \in \mathbb{B}_t^k$ $(i = 1,\ldots,k)$ and $\mathcal{P}_t^M$. The score is computed as (Equation 5.13).

$$s_{ti}^M = \frac{|\mathcal{P}_{ti} \cap \mathcal{P}_t^M|}{|\mathcal{P}_{ti} \cup \mathcal{P}_t^M|} \tag{5.13}$$

Note that s_{ti}^M measures the agreement between the bounding box $\mathbf{bb}_{ti}^+$ predicted by appearance component and the motion predicted target pixel positions $\mathcal{P}_t^M$. Let $\mathbb{S}_t^M = \{s_{ti}^M; i = 1,\ldots,k\}$ be the set of scores associated with the bounding boxes in $\mathbb{B}_t^k$. The set $\mathbb{B}_t^k$ is further refined by thresholding on the agreement between appearance and motion based predictions. This provides with the the refined set of bounding boxes $\mathbb{B}_t^R$ and their index set $\mathbf{J}_t$ (Equation 5.15).

$$\mathbb{B}_t^R = \{\mathbf{bb}_{ti}^+ \; : \; s_{ti}^M \geq s_{th}^M; i = 1,\ldots,k\} \tag{5.14}$$

$$\mathbf{J}_t = \{i \; : \; s_{ti}^M \geq s_{th}^M; i = 1,\ldots,k\} \tag{5.15}$$

Where, s_{th}^M is a predefined threshold. The contribution of motion component in the ensemble varies with the threshold s_{th}^M. The motion component is discarded if $\mathbb{B}_t^R = \varphi$. The following rule is used to finally localize the target object at $\overset{\star}{\mathbf{bb}}_t^+ := \mathbf{bb}_{ti^\star}^+$ in the current frame $\boldsymbol{\mathcal{I}}_t$.

$$i^\star = \begin{cases} \underset{i \in \mathbf{J}_t}{\operatorname{argmax}}\; s_{ti}^M \in \mathbb{S}_t^M, & \text{If } \mathbf{J}_t \neq \varphi \\ \underset{i=1,\ldots k}{\operatorname{argmax}}\; s_{ti} \in \mathbb{S}_t^k, & \text{Otherwise} \end{cases}$$

In case of absence of agreement between the two components, the ensemble relies on appearance component alone for tracking.

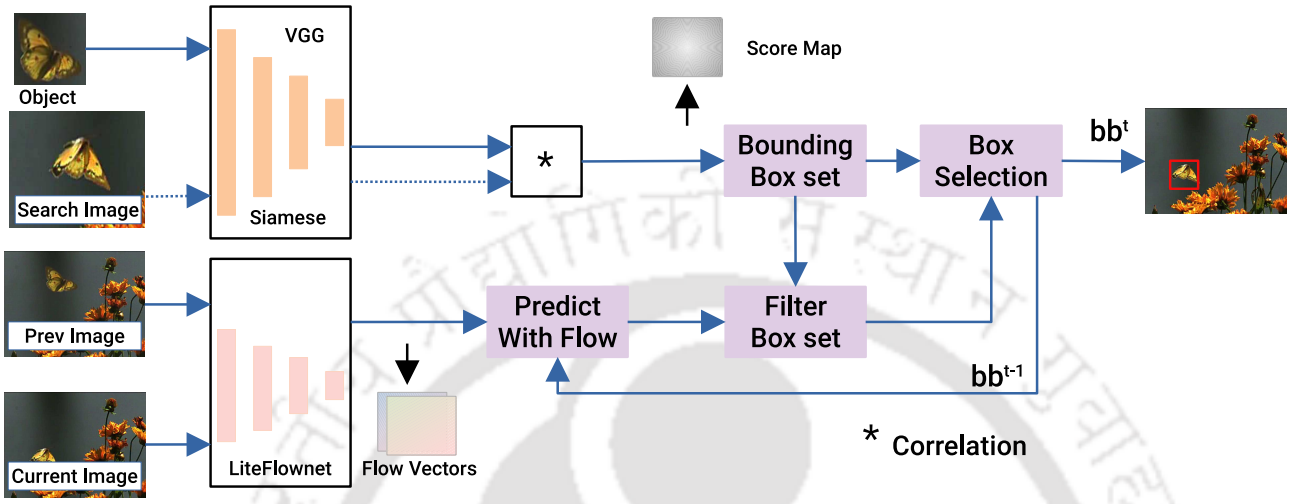

Figure 5.8: Proposed *SiamFC-PMC* . Multiple candidate boxes are returned by *SiamFC*. Pixel motion is predicted with *LiteFlownet*. The final bounding box $\mathbf{bb}_t^{\star+}$ is computed based on the overlap between prediction of two components.

Algorithms for the approaches based on single part (*SiamFC-MC*) and multiple part (*SiamFC-PMC*) models with motoin correction are presented in algorithms 5 and 4 respectively.

5.6 Experiments and Results

The proposed approaches are benchmarked on multiple datasets like $VOT2018$, $VOT2019$, $OTB2015$ and $UAV123$ 2.4. Evaluation paradigms corresponding to each dataset as provided in GOT-10k benchmarking tool is used for $VOT2019$, $OTB2015$ and $UAV123$ datasets. The VOT toolkit is used for the evaluation on $VOT2018$ dataset. The accuracy, robustness and expected average overlap values are evaluated for $VOT2018$ and $VOT2019$ datasets. The precision and success plots are evaluated for $OTB2015$ and $UAV123$ datasets.

5.6.1 Quantitative Analysis

Higher values of accuracy and expected average overlap (EAO) (↑) and lower value of robustness (↓) indicate better performance. The proposed tracker outperformed the baseline *SiamFC* tracker in terms of both accuracy and EAO. On $VOT2018$ dataset the proposed single part model *SiamFC-MC* outperformed the baseline tracker *SiamFC* in both accuracy (from $0.5002\ to\ 0.5121$) and expected

Algorithm 4 $SiamFC-PMC$

Input $\boldsymbol{\mathcal{I}}_0$, $\mathbf{bb}_0^+$

Output $\overset{\star}{\mathbf{bb}}{}_t^+$

$\mathbf{bb}_0^+ = [x_0, y_0, w_0, h_0]$

Define $\mathbf{bb}_{01}^+ = \left[x_0 - \frac{w_{(0)}}{4}, y_0 - \frac{h_0}{4}, \frac{w_0}{2}, \frac{h_0}{2}\right]$; $\boldsymbol{\mathcal{I}}_1^Q = extractROI(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_{01}^+)$

Define $\mathbf{bb}_{02}^+ = \left[x_0 + \frac{w_{(0)}}{4}, y_0 - \frac{h_0}{4}, \frac{w_0}{2}, \frac{h_0}{2}\right]$; $\boldsymbol{\mathcal{I}}_2^Q = extractROI(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_{02}^+)$

Define $\mathbf{bb}_{03}^+ = \left[x_0 - \frac{w_{(0)}}{4}, y_0 + \frac{h_0}{4}, \frac{w_0}{2}, \frac{h_0}{2}\right]$; $\boldsymbol{\mathcal{I}}_3^Q = extractROI(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_{03}^+)$

Define $\mathbf{bb}_{04}^+ = \left[x_0 + \frac{w_{(0)}}{4}, y_0 + \frac{h_0}{4}, \frac{w_0}{2}, \frac{h_0}{2}\right]$; $\boldsymbol{\mathcal{I}}_4^Q = extractROI(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_{04}^+)$

Define $\mathbf{bb}_{05}^+ = \left[x_0, y_0, \frac{w_0}{2}, \frac{h_0}{2}\right]$; $\boldsymbol{\mathcal{I}}_5^Q = extractROI(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_{05}^+)$

while $t \leq (n^T)$ **do** ▷ Loop through sequence

  $(\mathbf{U}_{(t-1)}, \mathbf{V}_{(t-1)}) = \phi_{LiteFlow}(\boldsymbol{\mathcal{I}}_{t-1}, \boldsymbol{\mathcal{I}}_t)$

  $\overset{\star}{\mathbf{bb}}{}_{(t-1)}^+ = [x_{(t-1)}, y_{(t-1)}, w_{(t-1)}, h_{(t-1)}]$

  $\boldsymbol{\mathcal{I}}_t^Q = extractROI(\boldsymbol{\mathcal{I}}_t, \overset{\star}{\mathbf{bb}}{}_{t-1}^+)$

  $i = 0$

  for $x \leftarrow x_{(t-1)} - \frac{w_{(t-1)}}{2}$ to $x_{(t-1)} + \frac{w_{(t-1)}}{2}$ **do**

    for $y \leftarrow y_{(t-1)} - \frac{h_{(t-1)}}{2}$ to $y_{(t-1)} + \frac{h_{(t-1)}}{2}$ **do**

      $\mathbf{bb}_{ti} = \left[x, y, \frac{w_{(t-1)}}{2}, \frac{h_{(t-1)}}{2}\right]$

      $\boldsymbol{\mathcal{I}}_{ti}^{\complement} = extractROI(\boldsymbol{\mathcal{I}}_t, \mathbf{bb}ti)$

      $\mathbb{S}_{ti1}(x,y) = \phi(\boldsymbol{\mathcal{I}}_1^Q, \boldsymbol{\mathcal{I}}_{ti}^{\complement})$, $\mathbb{S}_{ti2}(x,y) = \phi(\boldsymbol{\mathcal{I}}_2^Q, \boldsymbol{\mathcal{I}}_{ti}^{\complement})$, $\mathbb{S}_{ti3}(x,y) = \phi(\boldsymbol{\mathcal{I}}_3^Q, \boldsymbol{\mathcal{I}}_{ti}^{\complement})$,

      $\mathbb{S}_{ti4}(x,y) = \phi(\boldsymbol{\mathcal{I}}_4^Q, \boldsymbol{\mathcal{I}}_{ti}^{\complement})$, $\mathbb{S}_{ti5}(x,y) = \phi(\boldsymbol{\mathcal{I}}_5^Q, \boldsymbol{\mathcal{I}}_{ti}^{\complement})$

      $\mathbb{S}_{ti}(x,y) = w_1 \times \mathbb{S}_{ti1}(x - \frac{w_0}{2}, y - \frac{h_0}{2}) + w_2 \times \mathbb{S}_{ti2}(x + \frac{w_0}{2}, y - \frac{h_0}{2}) +$

$w_3 \times \mathbb{S}_{ti3}(x - \frac{w_0}{2}, y + \frac{h_0}{2}) + w_4 \times \mathbb{S}_{ti4}(x + \frac{w_0}{2}, y + \frac{h_0}{2}) + w_5 \times \mathbb{S}_{ti5}(x,y)$

      $\mathcal{P}_t^M = \{(x,y)_{(t-1)} + (\mathbf{U}_{t-1}[x,y], \mathbf{V}_{t-1}[x,y]), (x,y)_{t-1} \in \overset{\star}{\mathcal{P}}{}_i^{t-1}\}$

      $s_{ti}^M = \frac{|\mathcal{P}_t^M \cap \mathcal{P}_{ti}|}{|\mathcal{P}_t^M \cup \mathcal{P}_{ti}|}$, $i = i + 1$

    end for

  end for

  Generate the set $\mathbf{J}_t$.

  if $\mathbf{J}_t \neq \varphi$ **then**

    $\overset{\star}{\mathbf{bb}}{}_t^+ := \mathbf{bb}_{ti^\star}^+ \; ; i^\star = \underset{i \in \mathbf{J}_t}{\operatorname{argmax}}\, s_{ti}^M \in \mathbb{S}_t^M$

  else

    $\overset{\star}{\mathbf{bb}}{}_t^+ := \mathbf{bb}_{ti^\star}^+ \; ; i^\star = \underset{i=1,\ldots k}{\operatorname{argmax}} s_{ti}^M \in \mathbb{S}_t$

  end if

end while

Algorithm 5 $SiamFC-MC$

Input $\boldsymbol{\mathcal{I}}_0$, $\mathbf{bb}_0^+$

Output $\overset{\star}{\mathbf{bb}}{}_t^+$

Define $\boldsymbol{\mathcal{I}}^Q = extractROI(\boldsymbol{\mathcal{I}}_0, \mathbf{bb}_0^+)$
while $t \leq (n^T)$ **do** ▷ Loop through sequence
  $(\mathbf{U}_{(t-1)}, \mathbf{V}_{(t-1)}) = \phi_{LiteFlow}(\boldsymbol{\mathcal{I}}_{t-1}, \boldsymbol{\mathcal{I}}_t)$
  $\overset{\star}{\mathbf{bb}}{}_{(t-1)}^+ = [x_{(t-1)}, y_{(t-1)}, w_{(t-1)}, h_{(t-1)}]$
  $\boldsymbol{\mathcal{I}}_t^Q = extractROI(\boldsymbol{\mathcal{I}}_t, \overset{\star}{\mathbf{bb}}{}_{t-1}^+)$
  $i = 0$
  $\overset{\star}{\mathbf{bb}}{}_{t-}1^+ = [x_{t-1}, y_{t-1}, w_{t-1}, h_{t-1}]$
  for $x \leftarrow x_{(t-1)} - \frac{w_{(t-1)}}{2}$ to $x_{(t-1)} + \frac{w_{(t-1)}}{2}$ **do**
    for $y \leftarrow y_{(t-1)} - \frac{h_{(t-1)}}{2}$ to $y_{(t-1)} + \frac{h_{(t-1)}}{2}$ **do**
      $\mathbf{bb}_{ti} = \left[x + \frac{w_{(t-1)}}{2}, y + \frac{h_{(t-1)}}{2}, w_{(t-1)}, \frac{h_{(t-1)}}{2}\right]$
      $\boldsymbol{\mathcal{I}}_{ti}^{\complement} = extractROI(\boldsymbol{\mathcal{I}}_t, \mathbf{bb}ti), \mathbb{S}_{ti} = \phi(\boldsymbol{\mathcal{I}}^Q, \boldsymbol{\mathcal{I}}_{ti}^{\complement})$
      $\mathcal{P}_t^M = \{(x,y)_{(t-1)} + (\mathbf{U}_{t-1}[x,y], \mathbf{V}_{t-1}[x,y]), (x,y)_{t-1} \in \overset{\star}{\mathcal{P}}{}_i^{t-1}\}$
      $s_{ti}^M = \frac{|\mathcal{P}_t^M \cap \mathcal{P}_{ti}|}{|\mathcal{P}_t^M \cup \mathcal{P}_{ti}|}$, $i = i + 1$
    end for
  end for
  Generate the set $\mathbf{J}_t$.
  if $\mathbf{J}_t \neq \varphi$ **then**
    $\overset{\star}{\mathbf{bb}}{}_t^+ := \mathbf{bb}_{ti^\star}^+ ; i^\star = \underset{i \in \mathbf{J}_t}{\operatorname{argmax}}\, s_{ti}^M \in \mathbb{S}_t^M$
  else
    $\overset{\star}{\mathbf{bb}}{}_t^+ := \mathbf{bb}_{ti^\star}^+ ; i^\star = \underset{i=1,\ldots k}{\operatorname{argmax}} s_{ti}^M \in \mathbb{S}_t$
  end if
end while

average overlap (from 0.188 to 0.2013) but, with a degradation in robustness (from 2.0249 to 2.436). On $VOT2018$ dataset, the proposed multiple part model *SiamFC-PMC* outperformed the single part tracker *SiamFC-MC* in both accuracy (from 0.5121 to 0.5165) and EAO (from 0.2013 to 0.2161), but with reduced robustness (from 2.436 to 2.713). Performance comparison on $VOT2019$ dataset is provided in table 5.2. On $VOT2019$ dataset, proposed multiple part model *SiamFC-PMC* outperformed the single part tracker *SiamFC-MC* in accuracy (from 0.5093 to 0.5156), Robustness (from 53.6017 to 49.6676) and EAO (from 0.1845 to 0.1907).

The tracker performance was compared with 19 trackers available in $VOT2018$ and $VOT2019$ challenge [127, 211]. They include *ASMS* [71], *CSRDCF* [307], *DCFNet* [308], *DSiam* [127], *DeepC-SRDCF* [127], *DSST* [62], *ECO* [127], *FoT* [127], *KCF* [60], *L1APG* [31], *MBSiam* [127], *MIL* [36], *RAnet* [127], *SiamFC* [10], *SiamRPN* [144], *SiamVGG* [139], *SRDCF* [64], *Staple* [127], *UpdateNet* [127], *SiamMask* [149], *TADT* [200], *SSRCCOT* [211], *M2C2F* [211], *SiamRPNX* [211], *TDE* [211], *RSiamFC* [211], *RankingT* [211], *FSC2F* [211], *DPT* [211]. Most of the trackers compared are selected based on their state of the art performance and popularity. Some traditional tracking algorithms are also selected for comparison. Table 5.1 shows the comparison of the proposed trackers with benchmark trackers in $VOT2018$ challenge. Figures 5.9 and 5.10 shows the expected average overlap measure and accuracy robustness (Ψ_{AR}) respectively of different baseline trackers on $VOT2018$ dataset. Best trackers are to the top right corner of Ψ_{AR}. Plot 5.11 shows the variation in average IOU with the threshold.

The performance curves of the proposed tracker (*SiamFC-PMC*) on 100 sequences from $OTB2015$ dataset are shown in Figures 5.12 and 5.13. For comparison with other trackers on $OTB2015$ dataset, the plots are shown in Appendix A.2. Performance curves of *SiamFC-PMC* tracker on $UAV123$ dataset are shown in Figures 5.14 and 5.15.

5.6.2 Qualitative Analysis

Tracking outputs on some sequences from $VOT2018$ dataset are shown in Figure 5.16. The proposed tracker was able to adapt to scale changes of the object. Motion information helped in improving the performance as compared to the baseline tracker. Baseline *SiamFC* uses only single part appearance information for object tracking. Proposed modification with motion information showed improved performance on $VOT2018$ and $VOT2019$ datasets. These results are presented in Tables 5.1 and 5.2 respectively. The re-ranking strategy helped in selecting better candidate proposal resulting

	A-R rank		
	Accuracy(↑)	Robustness(↓)	EAO(↑)
ASMS	0.4884	2.250	0.1692
CSRDCF	0.4846	0.985	0.2561
DCFNet	0.4647	1.944	0.1825
DSiam	0.5089	2.507	0.1963
DeepCSRDCF	0.4827	0.985	0.2926
DSST	0.3895	5.430	0.0788
ECO	0.4758	1.117	0.2805
FoT	0.3901	4.190	0.1299
KCF	0.444	2.577	0.1349
L1APG	0.4209	8.066	0.0693
MBSiam	0.5242	1.795	0.2411
MIL	0.3847	3.933	0.1183
RAnet	0.4419	2.658	0.1415
SiamFC	0.5002	2.0249	0.188
SiamFC-PMC	***0.5165***	***2.713***	***0.2161***
SiamFC-MC	***0.5121***	***2.436***	***0.2013***
SiamRPN	0.5779	1.110	0.3827
SiamVGG	0.5254	1.193	0.2865
SRDCF	0.4802	3.31	0.1189
Staple	0.5244	2.507	0.1694
UpdateNet	0.509	1.771	0.2436

Table 5.1: Performance comparison on $VOT2018$ dataset with the baseline scheme in $VOT2018$ toolkit. Evaluation is done on 60 sequences available in the dataset. Experimental results show improvement over the baseline tracker. Proposed trackers are highlighted in bold and italics font. Evaluation is done with VOT toolkit.

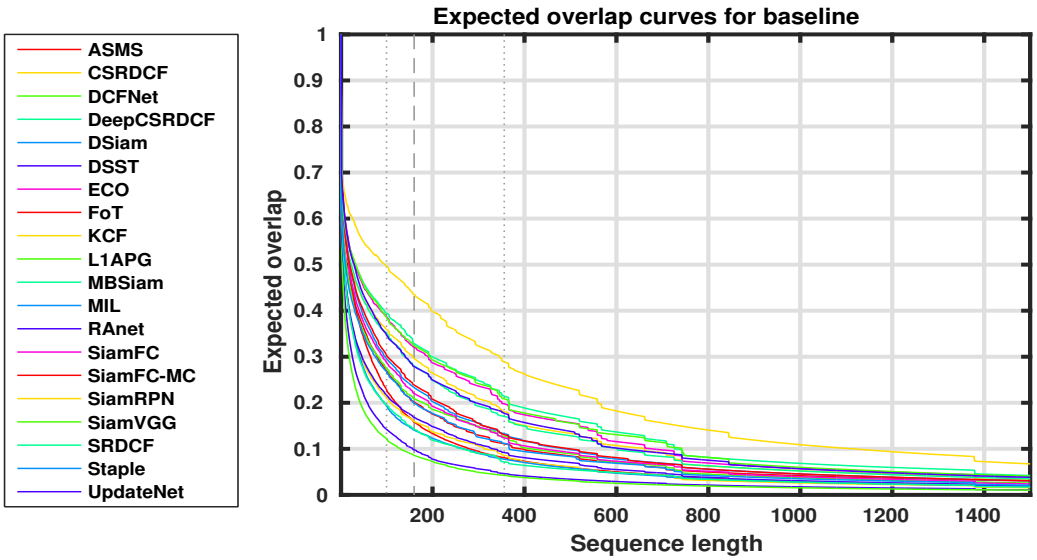

Figure 5.9: Expected overlap on baseline experiment of $VOT2018$ toolkit.

TH-3232_146102001

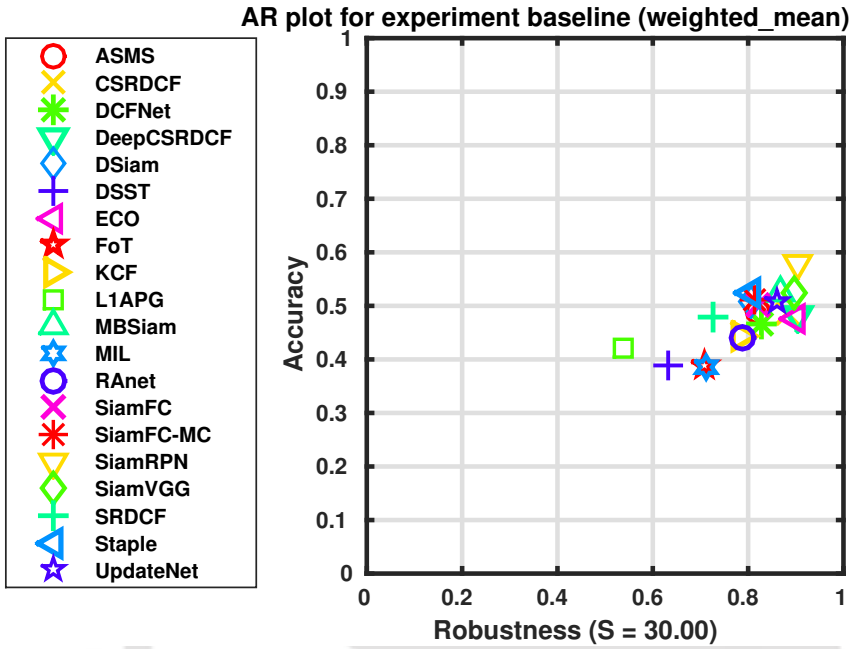

Figure 5.10: Accuracy robustness rank from $VOT2018$ toolkit. Better performing trackers are towards the right and top of the graph.

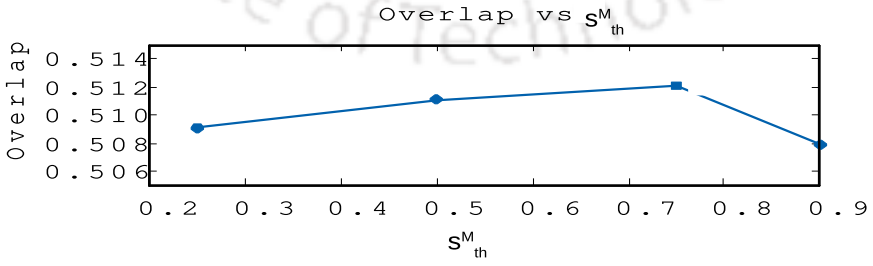

Figure 5.11: Performance variation with s^M_{th}.

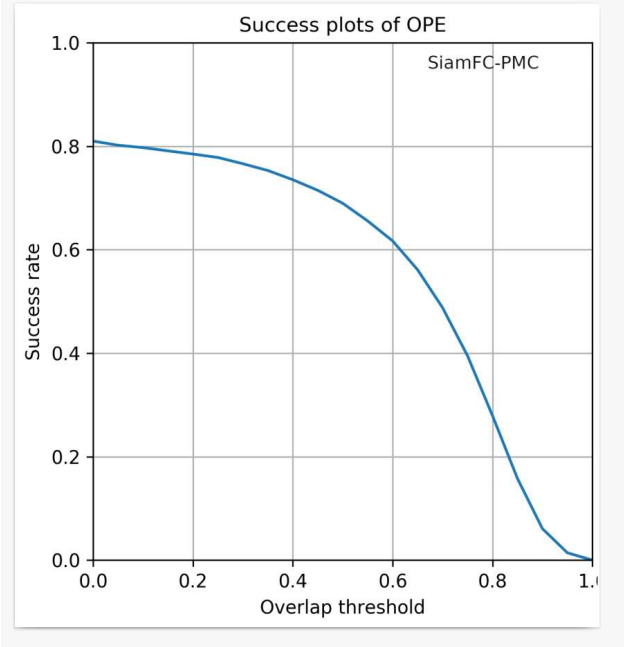

Figure 5.12: Overlap plot of siamese tracker on 100 sequences of $OTB2015$ dataset.

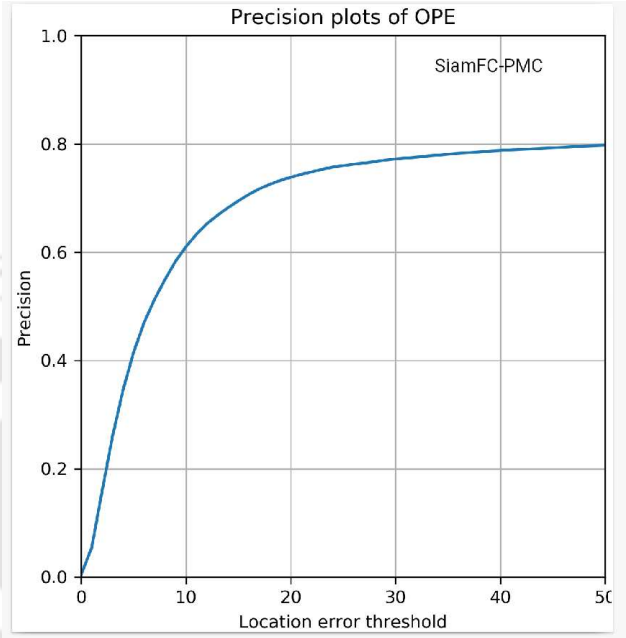

Figure 5.13: Precision plot of siamese tracker on 100 sequences of $OTB2015$ dataset

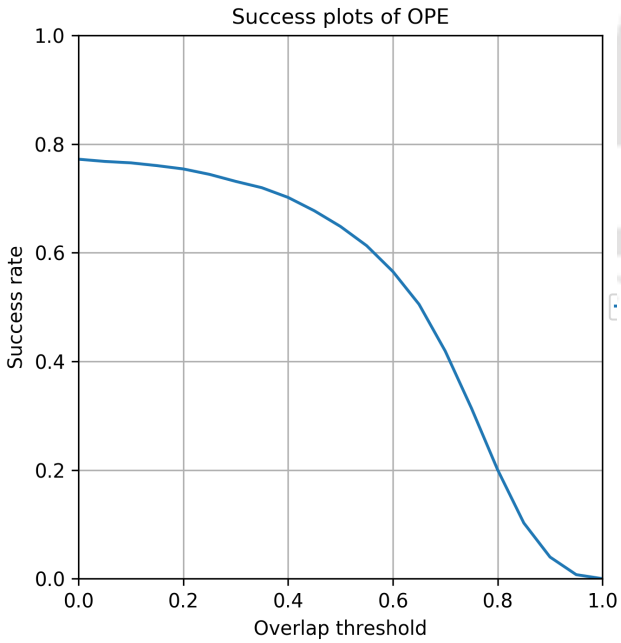

Figure 5.14: Success plot of siamese tracker on 123 sequences of $UAV123$ dataset.

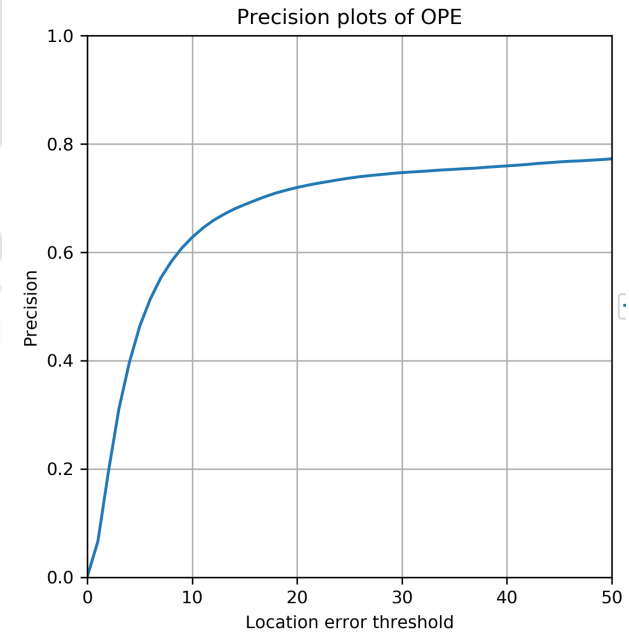

Figure 5.15: Precision plot of siamese tracker on 123 sequences of $UAV123$ dataset

	A-R rank		EAO
	Overlap	Failures	EAO
SiamFC-PMC	***0.5156***	***49.6676***	***0.1907***
SiamFC-MC	***0.5093***	***53.6017***	***0.1845***
SiamFCOSP [211]	0.5008	71.7348	0.1707
SiamMask [149]	0.5907	29.3580	0.2856
SiamRPNX [211]	0.5190	35.7638	0.2237
RSiamFC [211]	0.4659	58.7354	0.1627
NCC [211]	0.4636	173.0405	0.0846
CSRDCF [307]	0.4916	40.7859	0.2014
KCF [60]	0.4377	78.3864	0.1135
MIL [36]	0.3884	78.6186	0.1201
Struck [76]	0.4120	100.9718	0.0963
SSRCCOT [211]	0.4895	33.2756	0.2330
ASMS [71]	0.4731	55.2126	0.1595
FoT [127]	0.3643	77.7202	0.1290
M2C2F [211]	0.4770	47.1829	0.1767
L1APG [31]	0.3936	147.7971	0.0801
TADT [200]	0.5123	40.6218	0.2066
TDE [211]	0.5298	29.0514	0.2547
RankingT [211]	0.5241	22.5801	0.2698
DPT [211]	0.4832	60.1490	0.1587
FSC2F [211]	0.4711	48.5677	0.1850

Table 5.2: Performance comparison on $VOT2019$ dataset. Evaluation is performed on 60 sequences available in the dataset. Experimental results show reasonable performance with respect to other trackers. Top 3 performances are highlighted in red, blue and green respectively. Proposed trackers are highlighted in bold and italics font. Evaluation is performed with GOT-10K toolkit.

in performance improvement. The multiple part-based model further improved the performance by providing varied importance to different parts of the target object region.

One of the hyper-parameters to the algorithm is the threshold $s_{th}^{M} \in [0, 1]$ that controls the influence of motion on appearance component. A value of $s_{th}^{M} = 1$ nullifies the motion component and the tracker runs as an appearance only one. As the value of s_{th}^{M} is increased, it reduces the influence of motion information in the ensemble. The proposed tracker was adaptive to the scale changes of the object but, not to in-plane object rotations. Thus, the prediction output is always an upright (or axis-aligned) bounding box.

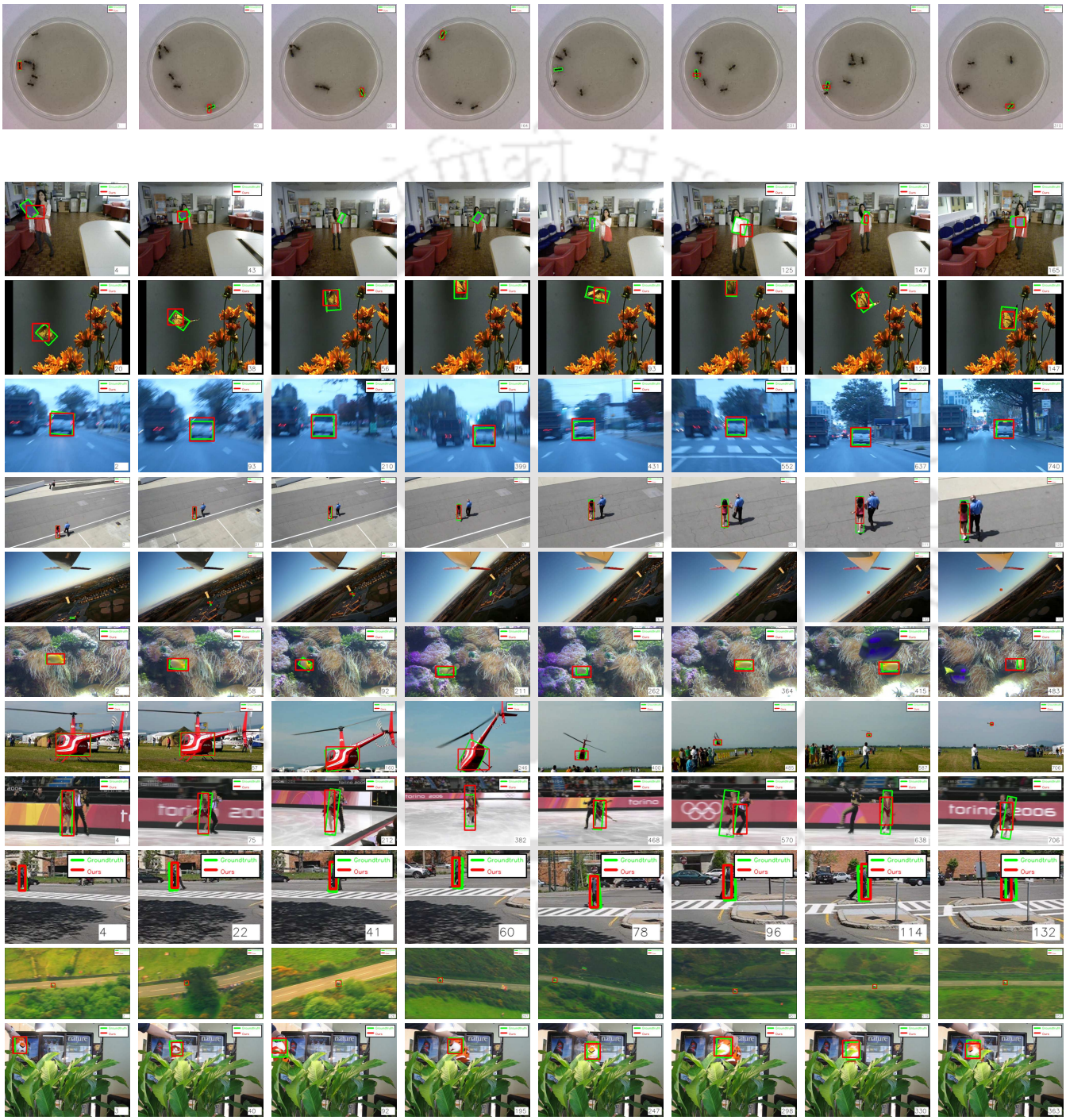

Figure 5.16: Each row shows the respective outputs of visual tracking on sequences from $VOT2018$ dataset. Groundtruth is marked in green and tracker prediction is colored red.

5.7 Discussions

In this chapter deep neural networks and their representational ability is explored to develop the tracking algorithm. Appearance feature based algorithm performed good among its peers. Proposed modification of inclusion of complementary motion improved trackers performance as highlighted by the experiments. Further improvement was brought in with the part-based object model. Proposed tracker with multi-part object model and complementary motion information outperformed the baseline tracker and showed competitive performance in comparison with other deep learning based trackers.

Proposed tacker always predicts an upright bounding box irrespective of the in-plane rotation of the object. A future extension can include prediction of oriented bounding box instead of upright ones. Currently, off-the-shelf models are used without further learning. Such an approach might not be optimized for the ensemble. Motion information can be embedded along with appearance to train a network in an end-to-end fashion. Such a multi-modal approach might improve overall robustness of the tracker against different challenges.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

6

Conclusion

Contents

Tracking algorithms belonging to both classical approaches and deep learning based techniques were explored in this thesis. The tracker proposals following the classical approaches have constructed models using distributions of color intensities and sparse codes. These models are learned for both object and its immediate background. The target localization was performed with a modified meta-heuristic search based on Firefly algorithm. Better tracking performance was achieved with deep features extracted from a siamese network. Here, the localization was performed in a brute force manner performed with the convolution operation. Effectiveness of motion information as a complementary component to appearance in an ensemble is explored. Experimental results support the use of motion information as a complementary component to appearance in tracking. The contributions of the thesis are briefly summarized next.

6.1 Summary

Chapter 3 has proposed the use of color distribution obtained from the target and its immediate background to construct the object model. Explicit modeling of immediate background of target helps in capturing contextual information. This is used in refining the object model as well as to define an objective function that searches for the target while also looking at the context. It can be observed from the experimental results that the inclusion of background information improves the tracker performance. Color distribution (obtained from entire object region) is a global feature and has limited spatial information of object appearance. The objective function with contextual component is used to overcome the impact of scaling due to the limited structural information as well as to improve robustness against presence of background information in target model. A meta-heuristic search strategy in breeding swarm framework was proposed for object localization. Extensive experiments were conducted to compare different evolutionary search and swarming technique based algorithms. Finally, a combination of Firefly Algorithm and Real-coded Genetic Algorithm is proposed. The incorporation of solution breeding (using RGA) into firefly population was observed to provide the best localization performance.

The object appearance can be represented by its colors and also from its structure. Arrangement of pixels in local neighborhoods decides the structure component of object appearance. Chapter 4 presents the second tracker proposal where the spare representation of vectorized image patches were explored to capture the structural information. First, the patch vectors are clustered to identify unique

patterns from the object and its immediate background. This is followed by dictionary learning with cluster specific atoms. The overall dictionary was defined from both object and background atoms to have the representational power for both. Object model in generative and hybrid approaches were proposed. The models were defined as distribution of sparse code coefficients. The hybrid approach had an additional classifier which down weighed the background information in the target model. Localization objective function was defined on the similarity between target and candidate models in the hybrid approach. In the generative model, explicit use of target model is avoided and the objective function was solely derived from the candidate model. Structural information was captured with the sparse coefficients. The target localization was performed by FRGA. However, some flexibility was lost as compared to color distribution based model. This was observed as lower performance of the sparse representation based tracker.

The tracker proposal presented in Chapter 5 explored the framework based on siamese networks. A better object appearance representation was obtained by using deep feature embeddings obtained from a pretrained convolutional neural network (here, VGG-16). The embedding(s) obtained from the target appearance is (are) compared (through correlation) against embeddings of candidate regions from within a search image region. This provides a correlation scoremap (localization confidence) for all possible candidate regions within the search image. This chapter explored both single and multiple part based object models. The multi-part object model was proposed to counter occlusions and positional weights were assigned to each object part. More weight was assigned to the central part (compared to peripheral ones) for handling partial occlusions. Experiments results showed that the siamese network based tracker proposal significantly outperformed the other two proposals.

Three tracker proposals were presented based on three different appearance features. These are (a) object color distribution (Chapter 3), (b) sparse code distribution of vectorized object patches (Chapter 4), and (c) deep network (VGG-16) embeddings of object parts (Chapter 5). Motion information is considered to be complementary with respect to appearance cues. Thus, the predictions of appearance based tracker and inter-frame dense motion were combined in an ensemble framework. The motion was estimated either by using classical optical flow techniques (Chapters 3 and 4) or by employing flow networks (Chapter 5). Experimental results demonstrated that motion based refinement led to performance improvement of all three proposed appearance based trackers. It was observed that, the multi-part siamese network based tracker with motion refinement provided the best

performance (Table 6.2). A comparison of the different approaches proposed is given in Table 6.1.

	Color Tracker (Chapter3)	Sparse Tracker (Chapter4)	Deep Tracker (Chapter5)
Features	Color intensity based and handcrafted	Learned from the limited data available in initial frame	Learned from large data and derived from a pre-trained model.
Localization Technique	Heuristic	Heuristic	Brute Force
Object Model Generalization	Good	Limited	Best

Table 6.1: Comparison of all the different approaches proposed.

The representational power of the sparsity-based approach and the deep network-based method differ on account of their access to training data. In a sparse framework, the training data is prepared from the first frame only. This restricts the availability of diversity in both positive (target object) and negative (background) class. On the other hand, the Siamese network-based approach uses pre-trained deep networks for generating image features. These deep networks are trained on large corpuses (for example, ImageNet dataset) for the task of image classification. Such datasets contain a large number of object categories with diverse scene background. This enables the pre-trained deep network to generate rich image representations. Such representations can handle the variability in the positive class as well as provide discrimination against other classes. This is also evident from the tracking results as the Siamese network-based tracker significantly outperforms the sparsity-based one.

A summary of performances of all the proposed trackers (on $VOT2018$ dataset) is presented in Table 6.2. The Accuracy-Robustness plot is shown in Figure 6.1 and the expected average overlap plot is shown in Figure 6.2. Visual output of all proposed trackers on some sequences from $VOT2018$ is shown in Figure 6.3.

From the experiments conducted on trackers belonging to classical and deep learning approaches, it is observed that deep feature embedding based trackers significantly outperform the classical ones. Also, use of motion as a complementary component to appearance aids in improving tracking performance.

Name	Accuracy ↑	Robustness ↓	EAO ↑
CWCD+FRGA	0.401	5.521	0.88
CWCD+FRGA+MOTION	0.397	5.245	0.086
DSC+FRGA	0.332	6.477	0.055
DSC+FRGA+MOTION	0.349	5.814	0.062
CWDSC+FRGA	0.351	6.396	0.055
CWDSC+FRGA+MOTION	0.359	5.640	0.066
SiamFC-MC	0.507	2.291	0.204
SiamFC-PMC	0.5165	2.713	0.2161

Table 6.2: Comparison of proposed trackers on $VOT2018$ dataset. Best, second best and third best are highlighted in red, blue and green respectively.

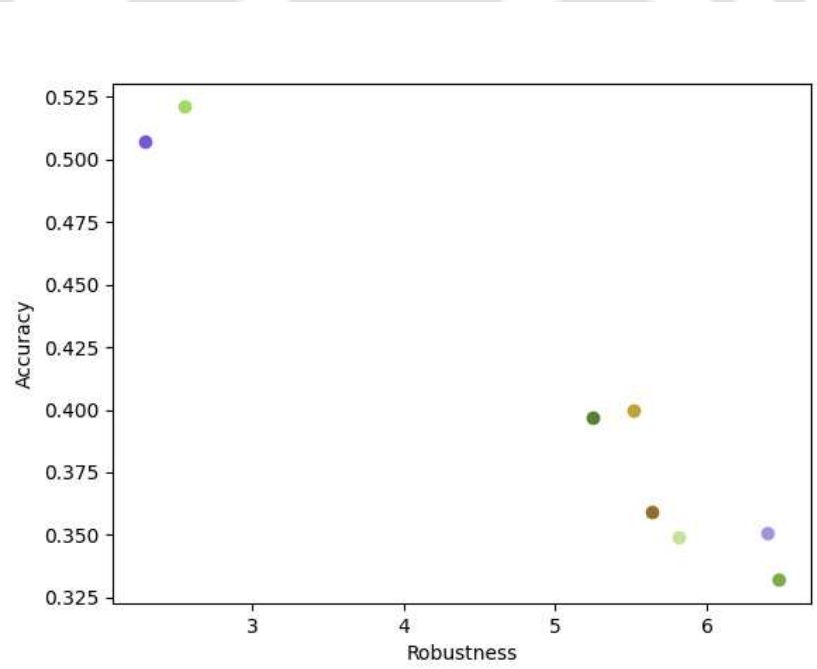

Figure 6.1: Accuracy-Robustness plot of tarckers *CWCD+FRGA* , *CWCD+FRGA+MOTION*,*DSC+FRGA*, *DSC+FRGA+MOTION*, *CWDSC+FRGA*, *CWDSC+FRGA+MOTION*, *SiamFC-MC*, *SiamFC-PMC*. *SiamFC-PMC* performs the best in both accuracy and EAO

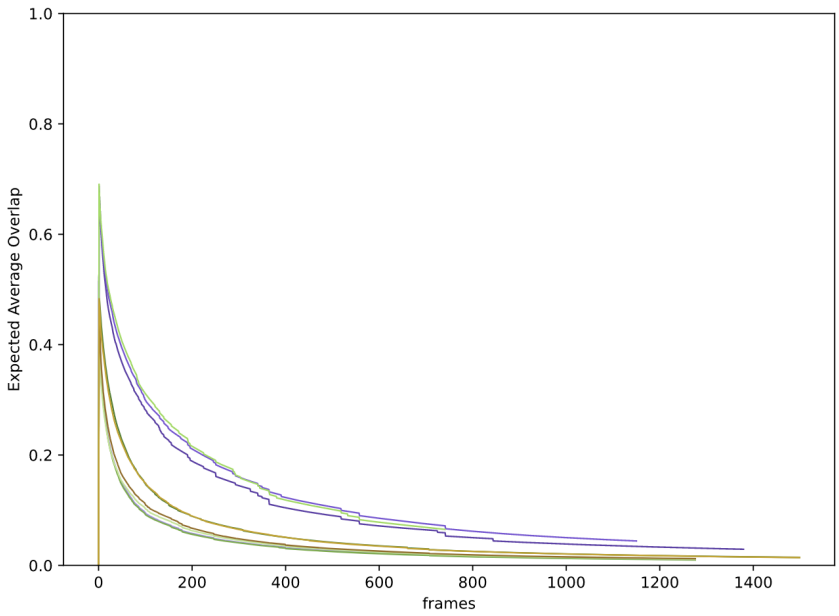

Figure 6.2: Expected average overlap plot of trackers *CWCD+FRGA* , *CWCD+FRGA+MOTION*,*DSC+FRGA*, *DSC+FRGA+MOTION*, *CWDSC+FRGA*, *CWDSC+FRGA+MOTION*, *SiamFC-MC*, *SiamFC-PMC*.

Figure 6.3: Each row shows the respective outputs of the proposed trackers on sequences from *VOT*2018 dataset. The ground-truth is marked in green and the predicted bounding boxes of proposed trackers CWCD+FRGA+MOTION, CWDSC+FRGA+MOTION, SIAMFC-PMC in different colors.

6.2 Future Work

It is hard to design a single tracking algorithm capable of handling all forms of tracking challenges. This can be concluded from existing works in the tracking literature and the observations from experiments conducted in this thesis. Based on observations from the experimental results of the three tracker proposals, the following are identified as potential future extensions of the present work.

- Appearance changes of the object and complex background create challenging scenarios for a tracker. A model update strategy is necessary to adapt to the appearance changes of the target as well as background. This is applicable to all the three tracker proposals. An automated component with a confidence score guided update of target and immediate background models may be explored [112]. Additionally, a scheme can be designed to learn optimal update parameters from data to minimize the chances of future tracker drift.

- Ensemble of trackers using complementary features is observed to provide improved performance. This work performed motion information based refinement of appearance tracker predictions. A potential extension to this work would be to learn the combiner. For example, networks can be learned from data to combine the spatio-temporal features or their predictions [309].

- Existing works in the tracking literature have proposed several approaches to counter specific challenges. For example, certain works have focused on illumination changes [310–312] only, while some attended to geometric deformations [61, 71, 149] or occlusions [15, 31, 313]. A multi-label classifier can be designed to identify the challenges in a set of frames. During operation, an ensemble of appropriate trackers can be deployed depending on the set of prevailing challenges [69, 107, 108] in a given frame.

- The siamese network based tracker explored in this thesis is limited to predicting only upright (or axis aligned) bounding boxes. This can be replaced with segmentation networks to obtain object mask predictions for better localization [149, 193, 208].

- Recent visual transformer based networks can be explored for defining better tracking algorithms [175, 205, 206, 209].

A

Benchmarks

Contents

A.1 $VOT2018$ Baseline Trackers

Name	A	R	EAO	Name	A	R	EAO
ALAL	0.5267	23.2	0.3889	**L1APG**	0.3961	91.8333	0.1946
ANT	0.4364	33.3333	0.3855	**LADCF**	0.4884	7.6667	0.1922
ASMS	0.474	29.1667	0.3836	**LGT**	0.3942	34.8556	0.1875
BDF	0.3668	54.8333	0.3784	**LSART**	0.4818	11.2	0.1872
BoVW CFT	0.476	22.8333	0.3761	**LWDNTm**	0.4442	17	0.1828
BST	0.2579	39.6222	0.3549	**LWDNTthi**	0.456	17.5	0.1733
C3DT	0.5111	23.8333	0.345	**Matflow**	0.3809	61	0.1713
CCOT	0.4643	17.6667	0.3388	**Matrioska**	0.3969	88.8333	0.1688
CFCF	0.4813	15.3333	0.3373	**MBSiam**	0.5101	21.1	0.1683
CFTR	0.4769	13.8333	0.3274	**MCCT**	0.5226	14.8333	0.1683
CPT fast	0.4995	15.5	0.2979	**MIL**	0.3675	50	0.1554
CSRDCF	0.4676	18.5	0.2964	**MRSNCC**	0.3184	68.5444	0.1543
csrtpp	0.4541	16.1667	0.2929	**PBTS**	0.3612	32.5556	0.1523
CSTEM	0.4447	22.3	0.2872	**RAnet**	0.424	35.3444	0.1443
DAT	0.4037	33.8333	0.2862	**RCO**	0.4982	8.1667	0.1442
DCFCF	0.4572	17.5	0.2808	**R MCPF**	0.4874	19.3333	0.1414
DCFNet	0.4632	28	0.2805	**RSECF**	0.458	25.1667	0.1385
DeepCSRDCF	0.4657	14.6667	0.2733	**SAPKLTF**	0.4727	29.3333	0.1377
DeepSTRCF1	0.5043	10.8333	0.2673	**SA Siam P**	0.5011	15.8333	0.1351
DensSiam	0.4477	37	0.2637	**SA Siam R**	0.5372	13.5	0.1301
DLSTpp	0.5359	12.0667	0.2609	**SiamRPN**	0.5684	14.6667	0.1179
DPT	0.4657	33	0.2576	**SiamVGG**	0.5177	16.1667	0.1176
DRT	0.4973	10.5	0.2568	**SRCT**	0.5115	15.5	0.116
DSiam	0.5037	32	0.2563	**SRDCF**	0.4794	46.8333	0.096
DSST	0.3904	67.8333	0.2477	**srdcf deep**	0.4752	34	0.0926
ECO	0.463	15.3333	0.2442	**srdcf dif**	0.4651	44.5	0.092
FoT	0.3697	50.5	0.2422	**Staple**	0.5134	31.8333	0.0822
FragTrack	0.3644	81.6667	0.2334	**STBACF**	0.4577	37.6667	0.0793
FSAN	0.5424	16.1889	0.2255	**struck2011**	0.4022	59.1667	0.0761
IVT	0.3787	75.1667	0.2097	**TRACA**	0.4411	40.8333	0.0695
KCF	0.4522	35.5	0.206	**UpdateNet**	0.4999	22.1667	0.0685

Table A.1: Results of trackers on $VOT2018$ dataset as provided by VOT challenge evaluated with GOT-10K toolkit .

A.2 $OTB2015$ Baseline Trackers

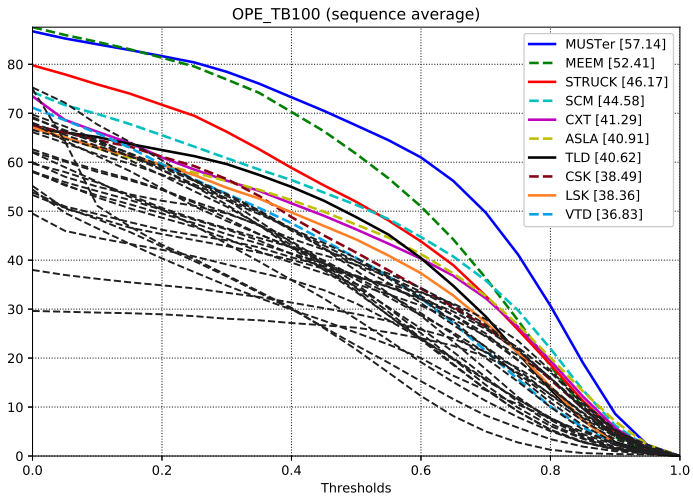

Figure A.1: Success plot of benchmark trackers on $OTB2015$ dataset.

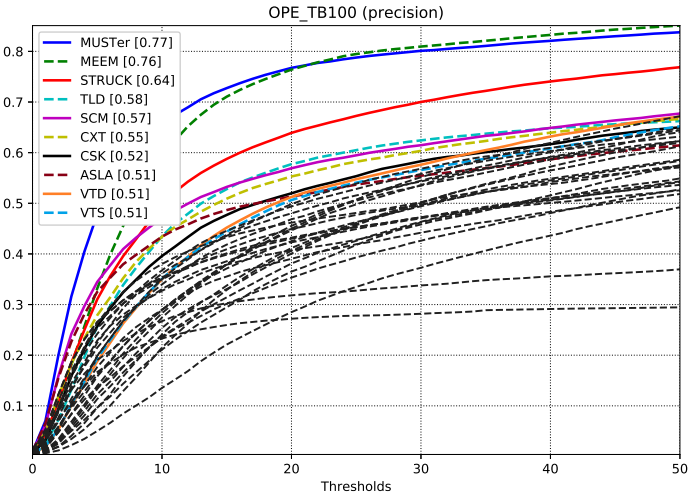

Figure A.2: Precision plot of benchmark trackers on $OTB2015$ dataset.

List of Publications

Conferences

1. M. Francis and P. Guha, 'Siamese Fully Convolutional Tracker with Motion Correction.' in International Conference on Pattern Recognition, 2020, pp. 2218–2225.
   (**Chapter - 5**)

2. P. Kate, M. Francis, and P. Guha, 'Visual Tracking with Breeding Fireflies using Brightness from Background-Foreground Information.' in International Conference on Pattern Recognition, 2018, pp. 2570–2575.
   (**Chapter - 3**)

3. M. Francis and P. Guha, 'Object Tracking with Classification Score Weighted Histogram of Sparse Codes.' in International Conference on Pattern Recognition and Machine Intelligence, 2017, pp. 162–169.
   (**Chapter - 4**)

4. R. Rajesh, M. Francis, and P. Guha, 'Tracking Under Scaling and Rotations Using Stochastic Mean Shift.' in India Council International Conference, 2015, pp. 1–6.

5. M. Francis, R. Ratnakaram, and P. Guha, 'PD-Shift: Patch Detector Shift based Tracker.' in National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics, 2015, pp. 1–4.

Bibliography

[1] A. W. M. 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.

[2] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A survey." *ACM Computing Surveys*, vol. 38, no. 4, p. 13, 2006.

[3] M. Ondrasovic and P. Tarábek, "Siamese Visual Object Tracking: A Survey." *IEEE Access*, vol. 9, pp. 110 149–110 172, 2021.

[4] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-Based Object Tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–575, 2003.

[5] L. Marchetti, G. Grisetti, and L. Iocchi, "A Comparative Analysis of Particle Filter Based Localization Methods." in *RoboCup*, ser. Lecture Notes in Computer Science, vol. 4434. Springer, 2006, pp. 442–449.

[6] S. Kok, D. N. Wilke, and A. A. Groenwold, "Recent Developments of the Particle Swarm Optimization Algorithm." in *Computational Intelligence*. IASTED/ACTA Press, 2005, pp. 392–397.

[7] R. C. Eberhart and Y. Shi, "Comparison between Genetic Algorithms and Particle Swarm Optimization." in *Evolutionary Programming*, ser. Lecture Notes in Computer Science, vol. 1447. Springer, 1998, pp. 611–616.

[8] X.-S. Yang, "Firefly Algorithms for Multimodal Optimization." in *Stochastic Algorithms: Foundations and Applications*, ser. Lecture Notes in Computer Science, vol. 5792. Springer, 2009, pp. 169–178.

[9] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real Time Robust L1 Tracker Using Accelerated Proximal Gradient Approach." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2012, pp. 1830–1837.

[10] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-Convolutional Siamese Networks for Object Tracking." in *European Conference on Computer Vision Workshops*, ser. Lecture Notes in Computer Science, vol. 9914, 2016, pp. 850–865.

[11] R. P. Pflugfelder, "Siamese Learning Visual Tracking: A Survey." *Computing Research Repository*, vol. abs/1707.00569, 2017.

[12] X. Lu, W. Wang, C. Ma, J. Shen, L. Shao, and F. Porikli, "See More, Know More: Unsupervised Video Object Segmentation with Co-Attention Siamese Networks." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2019, pp. 3623–3632.

[13] Y. Luo, Y. Cai, B. Wang, J. Wang, and Y. Wang, "SiamFF: Visual Tracking with a Siamese Network Combining Information Fusion with Rectangular Window Filtering." *IEEE Access*, vol. 8, pp. 119 899–119 910, 2020.

[14] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition." in *International Conference on Learning Representations*, 2015.

[15] Zhang, Tianzhu and Jia, Kui and Xu, Changsheng and Ma, Yi and Ahuja, Narendra, "Partial Occlusion Handling for Visual Tracking via Robust Part Matching." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2014, pp. 1258–1265.

[16] S. Kwak, W. Nam, B. Han, and J. H. Han, "Learning Occlusion with Likelihoods for Visual Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2011, pp. 1551–1558.

[17] X. Dong, J. Shen, D. Yu, W. Wang, J. Liu, and H. Huang, "Occlusion-Aware Real-Time Object Tracking." *IEEE Transactions on Multimedia*, vol. 19, no. 4, pp. 763–771, 2017.

[18] T.-W. Hui, X. Tang, and C. C. Loy, "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018, pp. 8981–8989.

[19] X. Li, W. Hu, C. Shen, Z. Zhang, A. R. Dick, and A. van den Hengel, "A Survey of Appearance Models in Visual Object Tracking." *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 4, pp. 58:1–58:48, 2013.

[20] F. Chen, X. Wang, Y. Zhao, S. Lv, and X. Niu, "Visual Object Tracking: A Survey." *Computer Vision and Image Understanding*, vol. 222, p. 103508, 2022.

[21] A. Blake and M. Isard, "The CONDENSATION Algorithm - Conditional Density Propagation and Applications to Visual Tracking." in *Conference on Neural Information Processing Systems*. MIT Press, 1996, pp. 361–367.

[22] K. Briechle and U. D. Hanebeck, "Template Matching Using Fast Normalized Cross Correlation." in *Proceedings of SPIE: Optical Pattern Recognition XII*, vol. 4387, Mar. 2001, pp. 95–102.

[23] X. Mei and H. Ling, "Robust Visual Tracking Using l1 Minimization." in *International Conference on Computer Vision*. IEEE Computer Society, 2009, pp. 1436–1443.

[24] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A Survey on New Generation Metaheuristic Algorithms," *Computers & Industrial Engineering*, vol. 137, 2019.

[25] L. Lv, T. Fan, Q. Li, Z. Sun, and L. Xu, "Object Tracking with Improved Firefly Algorithm." *International Journal of Computing Science and Mathematics*, vol. 9, no. 3, pp. 219–231, 2018.

[26] M.-L. Gao, X. He, D.-S. Luo, J. Jiang, and Q. Teng, "Object Tracking Using Firefly Algorithm." *IET Computer Vision*, vol. 7, no. 4, pp. 227–237, 2013.

[27] N. M. Artner and W. Burger, "A Comparison of Mean Shift Tracking Methods." in *12th Central European Seminar on computer graphics*, vol. 23, 2008.

[28] H. Zhou, Y. Yuan, and C. Shi, "Object Tracking Using SIFT Features and Mean Shift." *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 345–352, 2009.

[29] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: An Efficient Alternative to SIFT or SURF." in *International Conference on Computer Vision*. IEEE Computer Society, 2011, pp. 2564–2571.

[30] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast Retina Keypoint." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2012, pp. 510–517.

[31] X. Mei, H. Ling, Y. Wu, E. P. Blasch, and L. Bai, "Efficient Minimum Error Bounded Particle Resampling L1 Tracker with Occlusion Detection." *IEEE Transactions on Image Processing*, vol. 22, no. 7, pp. 2661–2675, 2013.

[32] N. Wang, J. Shi, D.-Y. Yeung, and J. Jia, "Understanding and Diagnosing Visual Tracking Systems." in *International Conference on Computer Vision*. IEEE Computer Society, 2015, pp. 3101–3109.

[33] H. T. Nguyen, M. Worring, and R. van den Boomgaard, "Occlusion Robust Adaptive Template Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2001, pp. 678–683.

[34] Y. Mae, Y. Shirai, J. Miura, and Y. Kuno, "Object Tracking in Cluttered Background Based on Optical Flow and Edges." in *International Conference on Pattern Recognition*. IEEE Computer Society, 1996, pp. 196–200.

[35] S. Avidan, "Support Vector Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2001, pp. 184–191.

[36] B. Babenko, M.-H. Yang, and S. J. Belongie, "Robust Object Tracking with Online Multiple Instance Learning." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.

[37] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[38] H.-K. Lee, K.-W. Choi, D. Kong, and J. Won, "Improved Kanade-Lucas-Tomasi Tracker for Images with Scale Changes." in *ICCE*. IEEE, 2013, pp. 33–34.

[39] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-Based Probabilistic Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 2350. Springer, 2002, pp. 661–675.

[40] J. Ning, L. Zhang, D. Zhang, and C. Wu, "Robust Mean-Shift Tracking with Corrected Background-Weighted Histogram." *IET Computer Vision*, vol. 6, no. 1, pp. 62–69, 2012.

[41] R. T. Collins, "Mean-shift Blob Tracking through Scale Space." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2003, pp. 234–240.

[42] Z. Zivkovic and B. J. A. Kröse, "An EM-Like Algorithm for Color-Histogram-Based Object Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 798–803.

[43] H. Possegger, T. Mauthner, and H. Bischof, "In Defense of Color-Based Model-Free Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2015, pp. 2113–2120.

[44] C. Yang, R. Duraiswami, and L. S. Davis, "Efficient Mean-Shift Tracking via a New Similarity Measure." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2005, pp. 176–183.

[45] T. Zhang, S. Liu, C. Xu, S. Yan, B. Ghanem, N. Ahuja, and M.-H. Yang, "Structural Sparse Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2015, pp. 150–158.

[46] B. Liu, J. Huang, C. A. Kulikowski, and L. Yang, "Robust Visual Tracking Using Local Sparse Appearance Model and K-Selection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2968–2981, 2013.

[47] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Low-Rank Sparse Learning for Robust Visual Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 7577. Springer, 2012, pp. 470–484.

[48] T. Bai and Y. Li, "Robust Visual Tracking Using Flexible Structured Sparse Representation." *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 538–547, 2014.

[49] Z. Chen, Z. Hong, and D. Tao, "An Experimental Survey on Correlation Filter-Based Tracking." *Computing Research Repository*, vol. abs/1509.05520, 2015.

[50] K. Zhang, L. Zhang, and M.-H. Yang, "Real-Time Compressive Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 7574. Springer, 2012, pp. 864–877.

[51] B. Babenko, M.-H. Yang, and S. J. Belongie, "Visual Tracking with Online Multiple Instance Learning." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2009, pp. 983–990.

[52] H. Grabner, M. Grabner, and H. Bischof, "Real-Time Tracking via On-line Boosting." in *British Machine Vision Conference*. British Machine Vision Association, 2006, pp. 47–56.

[53] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures." in *International Conference on Pattern Recognition*. IEEE Computer Society, 2010, pp. 2756–2759.

[54] 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. 1-3, pp. 125–141, 2008.

[55] H. Grabner, C. Leistner, and H. Bischof, "Semi-Supervised On-Line Boosting for Robust Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 5302. Springer, 2008, pp. 234–247.

[56] F. Tang, S. Brennan, Q. Zhao, and H. Tao, "Co-Tracking Using Semi-Supervised Support Vector Machines." in *International Conference on Computer Vision*. IEEE Computer Society, 2007, pp. 1–8.

[57] M. Godec, P. M. Roth, and H. Bischof, "Hough-Based Tracking of Non-Rigid Objects." in *International Conference on Computer Vision*. IEEE Computer Society, 2011, pp. 81–88.

[58] S. Duffner and C. Garcia, "PixelTrack: A Fast Adaptive Algorithm for Tracking Non-rigid Objects." in *International Conference on Computer Vision*. IEEE Computer Society, 2013, pp. 2480–2487.

[59] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual Object Tracking Using Adaptive Correlation Filters." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2010, pp. 2544–2550.

[60] 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.

[61] Y. Li and J. Zhu, "A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration." in *European Conference on Computer Vision Workshops*, ser. Lecture Notes in Computer Science, vol. 8926. Springer, 2014, pp. 254–265.

[62] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Discriminative Scale Space Tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1561–1575, 2017.

[63] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-Term Correlation Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2015, pp. 5388–5396.

[64] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Learning Spatially Regularized Correlation Filters for Visual Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2015, pp. 4310–4318.

[65] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang, "Learning Spatial-Temporal Regularized Correlation Filters for Visual Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018, pp. 4904–4913.

[66] M. Mueller, N. Smith, and B. Ghanem, "Context-Aware Correlation Filter Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017, pp. 1387–1395.

[67] A. Bibi, M. Mueller, and B. Ghanem, "Target Response Adaptation for Correlation Filter Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 9910. Springer, 2016, pp. 419–433.

[68] Z. Hong, Z. Chen, C. Wang, X. Mei, D. V. Prokhorov, and D. Tao, "MUlti-Store Tracker (MUSTer): A Cognitive Psychology Inspired Approach to Object Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2015, pp. 749–758.

[69] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr, "Staple: Complementary Learners for Real-Time Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2016, pp. 1401–1409.

[70] A. Lukezic, T. Vojír, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative Correlation Filter Tracker with Channel and Spatial Reliability." *International Journal of Computer Vision*, vol. 126, no. 7, pp. 671–688, 2018.

[71] T. Vojir, J. Noskova, and J. Matas, "Robust Scale-Adaptive Mean-Shift for Tracking." *Pattern Recognition Letters*, vol. 49, pp. 250–258, 2014.

[72] B. Liu, J. Huang, L. Yang, and C. A. Kulikowski, "Robust Tracking Using Local Sparse Appearance Model and K-Selection." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2011, pp. 1313–1320.

[73] T. Bai and Y. Li, "Robust Visual Tracking with Structured Sparse Representation Appearance Model." *Pattern Recognition*, vol. 45, no. 6, pp. 2390–2404, 2012.

[74] J.-Y. Bouguet *et al.*, "Pyramidal Implementation of the Affine Lucas Kanade Feature Tracker Description of the Algorithm." *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001.

[75] S. Avidan, "Ensemble Tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 261–271, 2007.

[76] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. S. Torr, "Struck: Structured Output Tracking with Kernels." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[77] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof, "PROST: Parallel Robust Online Simple Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2010, pp. 723–730.

[78] I. A. Matthews, T. Ishikawa, and S. Baker, "The Template Update Problem." in *British Machine Vision Conference*. BMVA Press, 2003, pp. 1–10.

[79] F. Porikli, O. Tuzel, and P. Meer, "Covariance Tracking using Model Update Based on Lie Algebra." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2006, pp. 728–735.

[80] S. Gladh, M. Danelljan, F. S. Khan, and M. Felsberg, "Deep Motion Features for Visual Tracking." in *International Conference on Pattern Recognition*. IEEE Computer Society, 2016, pp. 1243–1248.

[81] M.-L. Gao, L.-L. Li, X.-M. Sun, L.-J. Yin, H.-T. Li, and D.-S. Luo, "Firefly Algorithm (FA) Based particle Filter Method for Visual Tracking." *Optik*, vol. 126, no. 18, pp. 1705–1711, 2015.

[82] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[83] M. Isard and A. Blake, "CONDENSATION - Conditional Density Propagation for Visual Tracking." *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[84] S. S. Moghaddasi and N. Faraji, "A Hybrid Algorithm based on Particle Filter and Genetic Algorithm for Target Tracking." *Expert Systems with Applications*, vol. 147, p. 113188, 2020.

[85] R. C. Eberhart and Y. Shi, "Particle Swarm Optimization: Developments, Applications and Resources." in *Proceedings of the 2001 Congress on Evolutionary Computation, CEC 2001, COEX, Seoul, Korea, May 27-30, 2001*. IEEE, 2001, pp. 81–86.

[86] X.-S. Yang and X. He,"Firefly Algorithm: Recent Advances and Applications." *Computing Research Repository*, vol. abs/1308.3898, 2013.

[87] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: Robust Tracking via Multiple Experts Using Entropy Minimization." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 8694. Springer, 2014, pp. 188–203.

[88] G. Zhu, F. Porikli, and H. Li, "Tracking Randomly Moving Objects on Edge Box Proposals." *Computing Research Repository*, vol. abs/1507.08085, 2015.

[89] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow." *Artificial Intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[90] R. Caruana and J. D. Schaffer, "Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms." in *ML*. Morgan Kaufmann, 1988, pp. 153–161.

[91] M. Zhai, X. Xiang, N. Lv, and X. Kong, "Optical Flow and Scene Flow Estimation: A Survey." *Pattern Recognition*, vol. 114, p. 107861, 2021.

[92] 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.

[93] W. Enkelmann, "Investigations of Multigrid Algorithms for the Estimation of Optical Flow Fields in Image Sequences." *Computer Vision, Graphics, and Image Processing*, vol. 43, no. 2, pp. 150–177, 1988.

[94] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion." in *SCIA*, ser. Lecture Notes in Computer Science, vol. 2749. Springer, 2003, pp. 363–370.

[95] S. Han, M.-j. Kim, V. Maik, J. Paik *et al.*, "Keypoint-Based Object Tracking Using Modified Median Flow." in *IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, 2016, pp. 1–2.

[96] R. Okada, Y. Shirai, and J. Miura, "Object Tracking Based on Optical Flow and Depth." in *IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems (Cat. No. 96TH8242)*. IEEE, 1996, pp. 565–571.

[97] S. Shantaiya, K. Verma, and K. Mehta, "Multiple Object Tracking Using Kalman Filter and Optical Flow." *European Journal of Advances in Engineering and Technology*, vol. 2, no. 2, pp. 34–39, 2015.

[98] S. Yamamoto, Y. Mae, Y. Shirai, and J. Miura, "Realtime Multiple Object Tracking Based on Optical Flows," in *Proceedings of the 1995 International Conference on Robotics and Automation.* IEEE Computer Society, 1995, pp. 2328–2333.

[99] E. Alpaydin, *Introduction to Machine Learning.*, ser. Adaptive Computation and Machine Learning. MIT Press, 2004.

[100] Y. Ren, L. Zhang, and P. N. Suganthan, "Ensemble Classification and Regression-Recent Developments, Applications and Future Directions [Review Article]." *IEEE Computational Intelligence Magazine*, vol. 11, no. 1, pp. 41–53, 2016.

[101] M. Kristan, J. Matas, A. Leonardis, T. Vojír, R. P. Pflugfelder, G. Fernández, G. Nebehay, F. Porikli, and L. Cehovin, "A Novel Performance Evaluation Methodology for Single-Target Trackers." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2137–2155, 2016.

[102] A. Adam, E. Rivlin, and I. Shimshoni, "Robust Fragments-Based Tracking Using the Integral Histogram." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2006, pp. 798–805.

[103] J. Kwon and K. M. Lee, "Visual Tracking Decomposition." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2010, pp. 1269–1276.

[104] M. Tian, W. Zhang, and F. Liu, "On-Line Ensemble SVM for Robust Object Tracking." in *Asian Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 4843. Springer, 2007, pp. 355–364.

[105] J. Kwon and K. M. Lee, "Tracking by Sampling Trackers." in *International Conference on Computer Vision*. IEEE Computer Society, 2011, pp. 1195–1202.

[106] P. Senna, I. N. Drummond, and G. S. Bastos, "Real-Time Ensemble-Based Tracker with Kalman Filter." in *SIBGRAPI*. IEEE Computer Society, 2017, pp. 338–344.

[107] Z. Zhou, J. Chen, W. Pei, K. Mao, H. Wang, and Z. He, "Global Tracking via Ensemble of Local Trackers." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8761–8770.

[108] Y. Gao, R. Ji, L. Zhang, and A. G. 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.

[109] G. R. Bradski, "Real Time Face and Object Tracking as a Component of a Perceptual User Interface." in *IEEE Winter Conference on Applications of Computer Vision*. IEEE Computer Society, 1998, pp. 214–219.

[110] D. Comaniciu, V. Ramesh, and P. Meer, "Real-Time Tracking of Non-Rigid Objects Using Mean Shift," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2000, p. 2142.

[111] C. Stauffer and W. E. L. Grimson, "Learning Patterns of Activity Using Real-Time Tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, 2000.

[112] I. A. Matthews, T. Ishikawa, and S. Baker, "The Template Update Problem." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 810–815, 2004.

[113] R. T. Collins, Y. Liu, and M. Leordeanu, "Online Selection of Discriminative Tracking Features."*IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1631–1643, 2005.

[114] W. Zhong, H. Lu, and M.-H. Yang, "Robust Object Tracking via Sparsity-Based Collaborative Model." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2012, pp. 1838–1845.

[115] J. F. Henriques, R. Caseiro, P. Martins, and J. P. Batista, "Exploiting the Circulant Structure of Tracking-by-Detection with Kernels." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 7575. Springer, 2012, pp. 702–715.

[116] X. Jia, H. Lu, and M.-H. Yang, "Visual Tracking via Adaptive Structural Local Sparse Appearance Model." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2012, pp. 1822–1829.

[117] Y. Wu, J. Lim, and M.-H. Yang, "Online Object Tracking: A Benchmark." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2013, pp. 2411–2418.

[118] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Accurate Scale Estimation for Robust Visual Tracking." in *British Machine Vision Conference*. BMVA Press, 2014.

[119] M. Danelljan, F. S. Khan, M. Felsberg, and J. van de Weijer, "Adaptive Color Attributes for Real-Time Visual Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2014, pp. 1090–1097.

[120] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical Convolutional Features for Visual Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2015, pp. 3074–3082.

[121] K. Fukushima, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position." *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980.

[122] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition." *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[123] G. E. Hinton, S. Osindero, and Y. W. Teh, "A Fast Learning Algorithm for Deep Belief Nets." *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[124] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks." in *Conference on Neural Information Processing Systems*, 2012, pp. 1106–1114.

[125] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning." *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[126] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang, "Hedged Deep Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2016, pp. 4303–4311.

[127] Kristan, Matej and Leonardis, Ales and Matas, et.al , "The Sixth Visual Object Tracking VOT2018 Challenge Results." in *European Conference on Computer Vision Workshops*, ser. Lecture Notes in Computer Science, vol. 11129. Springer, 2018, pp. 3–53.

[128] R. Pflugfelder, "An In-Depth Analysis of Visual Tracking with Siamese Neural Networks." *arXiv preprint arXiv:1707.00569*, 2017.

[129] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2015, pp. 815–823.

[130] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature Verification Using a Siamese Time Delay Neural Network." in *Conference on Neural Information Processing*. Morgan Kaufmann, 1993, pp. 737–744.

[131] A. He, C. Luo, X. Tian, and W. Zeng, "A Twofold Siamese Network for Real-Time Object Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018, pp. 4834–4843.

[132] S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, and S. Kasaei, "Deep Learning for Visual Tracking: A Comprehensive Survey." *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 5, pp. 3943–3968, 2022.

[133] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi, "Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017, pp. 1349–1358.

[134] S. Javed, M. Danelljan, F. S. Khan, M. H. Khan, M. Felsberg, and J. Matas, "Visual Object Tracking With Discriminative Filters and Siamese Networks: A Survey and Outlook." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 6552–6574, 2023.

[135] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully Convolutional One-Stage Object Detection." in *International Conference on Computer Vision*. IEEE Computer Society, 2019, pp. 9626–9635.

[136] I. Sosnovik, A. Moskalev, and A. W. M. Smeulders, "Scale Equivariance Improves Siamese Tracking." in *IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2021, pp. 2764–2773.

[137] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese Instance Search for Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2016, pp. 1420–1429.

[138] D. Held, S. Thrun, and S. Savarese, "Learning to Track at 100 FPS with Deep Regression Networks." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 9905. Springer, 2016, pp. 749–765.

[139] Y. Li and X. Zhang, "SiamVGG: Visual Tracking using Deeper Siamese Networks." *Computing Research Repository*, vol. abs/1902.02804, 2019.

[140] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2016, pp. 770–778.

[141] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions." *Computing Research Repository*, vol. abs/1409.4842, 2014.

[142] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2019, pp. 4282–4291.

[143] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung, "Transferring Rich Feature Hierarchies for Robust Visual Tracking." *Computing Research Repository*, vol. abs/1501.04587, 2015.

[144] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High Performance Visual Tracking with Siamese Region Proposal Network." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018, pp. 8971–8980.

[145] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." in *Conference on Neural Information Processing Systems*, 2015, pp. 91–99.

[146] D. Guo, J. Wang, Y. Cui, Z. Wang, and S. Chen, "SiamCAR: Siamese Fully Convolutional Classification and Regression for Visual Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2020, pp. 6268–6276.

[147] Z. Chen, B. Zhong, G. Li, S. Zhang, and R. Ji, "Siamese Box Adaptive Network for Visual Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2020, pp. 6667–6676.

[148] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning Dynamic Siamese Network for Visual Object Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2017, pp. 1781–1789.

[149] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast Online Object Tracking and Segmentation: A Unifying Approach." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2019, pp. 1328–1338.

[150] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-Aware Siamese Networks for Visual Object Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 11213. Springer, 2018, pp. 103–119.

[151] D. Li, Y. Yu, and X. Chen, "Object tracking framework with Siamese Network and Re-Detection Mechanism." *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, p. 261, 2019.

[152] P. Voigtlaender, J. Luiten, P. H. S. Torr, and B. Leibe, "Siam R-CNN: Visual Tracking by Re-Detection." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2020, pp. 6577–6587.

[153] Z. Liang and J. Shen, "Local Semantic Siamese Networks for Fast Tracking." *IEEE Transactions on Image Processing*, vol. 29, pp. 3351–3364, 2020.

[154] F. Wu, J. Zhang, and Z. Xu, "Stably Adaptive Anti-Occlusion Siamese Region Proposal Network for Real-Time Object Tracking." *IEEE Access*, vol. 8, pp. 161 349–161 360, 2020.

[155] D. Li and Y. Yu, "Foreground Information Guidance for Siamese Visual Tracking." *IEEE Access*, vol. 8, pp. 55 905–55 914, 2020.

[156] 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*, ser. Lecture Notes in Computer Science, vol. 9909. Springer, 2016, pp. 472–488.

[157] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient Convolution Operators for Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017, pp. 6931–6939.

[158] Y. Li, C. Fu, F. Ding, Z. Huang, and G. Lu, "AutoTrack: Towards High-Performance Visual Tracking for UAV with Automatic Spatio-Temporal Regularization." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2020, pp. 11 920–11 929.

[159] T. Xu, Z.-H. Feng, X.-J. Wu, and J. Kittler, "Joint Group Feature Selection and Discriminative Filter Learning for Robust Visual Object Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2019, pp. 7949–7959.

[160] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ATOM: Accurate Tracking by Overlap Maximization." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2019, pp. 4660–4669.

[161] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Learning Discriminative Model Prediction for Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2019, pp. 6181–6190.

[162] M. Fiaz, A. Mahmood, S. Javed, and S. K. Jung, "Handcrafted and Deep Trackers: Recent Visual Object Tracking Approaches and Trends." *ACM Computing Surveys*, vol. 52, no. 2, pp. 43:1–43:44, 2019.

[163] Q. Gan, Q. Guo, Z. Zhang, and K. Cho, “First Step toward Model-Free, Anonymous Object Tracking with Recurrent Neural Network.” *Computing Research Repository*, vol. abs/1511.06425, 2015.

[164] H. Fan and H. Ling, “SANet: Structure-Aware Network for Visual Tracking.” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE Computer Society, 2017, pp. 2217–2224.

[165] G. Ning, Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He,“Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking.” in *ISCAS*. IEEE, 2017, pp. 1–4.

[166] Z. Cui, S. Xiao, J. Feng, and S. Yan, “Recurrently Target-Attending Tracking.” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2016, pp. 1449–1458.

[167] A. R. Kosiorek, A. Bewley, and I. Posner, “Hierarchical Attentive Recurrent Tracking.” in *Conference on Neural Information Processing Systems*, 2017, pp. 3053–3061.

[168] S. E. Kahou, V. Michalski, R. Memisevic, C. J. Pal, and P. Vincent, “RATM: Recurrent Attentive Tracking Model.” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. IEEE Computer Society, 2017, pp. 1613–1622.

[169] D. Gordon, A. Farhadi, and D. Fox, “Re3 : Real-Time Recurrent Regression Networks for Object Tracking.” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 788–795, 2018.

[170] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi,“You Only Look Once: Unified, Real-Time Object Detection.” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2016, pp. 779–788.

[171] G. Bhat, F. J. Lawin, M. Danelljan, A. Robinson, M. Felsberg, L. V. Gool, and R. Timofte, “Learning What to Learn for Video Object Segmentation.” in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 12347. Springer, 2020, pp. 777–794.

[172] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is All you Need.” in *Conference on Neural Information Processing Systems*, 2017, pp. 5998–6008.

[173] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.” in *ICLR*. OpenReview.net, 2021.

[174] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-End Object Detection with Transformers.” in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 12346. Springer, 2020, pp. 213–229.

[175] L. Lin, H. Fan, Z. Zhang, Y. Xu, and H. Ling,"SwinTrack: A Simple and Strong Baseline for Transformer Tracking." in *Conference on Neural Information Processing Systems*, 2022.

[176] N. Wang, W. Zhou, J. Wang, and H. Li, "Transformer Meets Tracker: Exploiting Temporal Context for Robust Visual Tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2021, pp. 1571–1580.

[177] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2021, pp. 8126–8135.

[178] C. Mayer, M. Danelljan, D. P. Paudel, and L. V. Gool, "Learning Target Candidate Association to Keep Track of What Not to Track." in *International Conference on Computer Vision*. IEEE Computer Society, 2021, pp. 13 424–13 434.

[179] C. Huang, S. Lucey, and D. Ramanan, "Learning Policies for Adaptive Tracking with Deep Feature Cascades." in *International Conference on Computer Vision*. IEEE Computer Society, 2017, pp. 105–114.

[180] L. Ren, X. Yuan, J. Lu, M. Yang, and J. Zhou, "Deep Reinforcement Learning with Iterative Shift for Visual Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 11213. Springer, 2018, pp. 697–713.

[181] J. S. S. III and D. Ramanan, "Tracking as Online Decision-Making: Learning a Policy from Streaming Videos with Reinforcement Learning." in *International Conference on Computer Vision*. IEEE Computer Society, 2017, pp. 322–331.

[182] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang, "Deep Reinforcement Learning for Visual Object Tracking in Videos." *Computing Research Repository*, vol. abs/1701.08936, 2017.

[183] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. H. Lau, and M.-H. Yang, "VITAL: Visual Tracking via Adversarial Learning." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018, pp. 8990–8999.

[184] B. Chen, D. Wang, P. Li, S. Wang, and H. Lu, "Real-Time 'Actor-Critic' Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 11211. Springer, 2018, pp. 328–345.

[185] M. Sun, J. Xiao, E. G. Lim, B. Zhang, and Y. Zhao, "Fast Template Matching and Update for Video Object Tracking and Segmentation." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2020, pp. 10 788–10 796.

[186] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative Adversarial Nets." in *Conference on Neural Information Processing Systems*, 2014, pp. 2672–2680.

[187] L. Huang, X. Zhao, and K. Huang, "Bridging the Gap Between Detection and Tracking: A Unified Approach." in *International Conference on Computer Vision*. IEEE Computer Society, 2019, pp. 3998–4008.

[188] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." in *International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 1126–1135.

[189] G. Wang, C. Luo, X. Sun, Z. Xiong, and W. Zeng,"Tracking by Instance Detection: A Meta-Learning Approach." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2020, pp. 6287–6296.

[190] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang, "Video Tracking Using Learned Hierarchical Features." *IEEE Transactions on Image Processing*, vol. 24, no. 4, pp. 1424–1435, 2015.

[191] H. Nam, M. Baek, and B. Han, "Modeling and Propagating CNNs in a Tree Structure for Visual Tracking." *Computing Research Repository*, vol. abs/1608.07242, 2016.

[192] Z. Teng, J. Xing, Q. Wang, C. Lang, S. Feng, and Y. Jin, "Robust Object Tracking Based on Temporal and Spatial Deep Networks." in *International Conference on Computer Vision*. IEEE Computer Society, 2017, pp. 1153–1162.

[193] A. Lukezic, J. Matas, and M. Kristan, "D3S - A Discriminative Single Shot Segmentation Tracker." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2020, pp. 7131–7140.

[194] M. Dunnhofer, N. Martinel, and C. Micheloni, "Tracking-by-Trackers with a Distilled and Reinforced Model." in *Asian Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 12623. Springer, 2020, pp. 631–650.

[195] P. Li, D. Wang, L. Wang, and H. Lu, "Deep Visual Tracking: Review and Experimental Comparison." *Pattern Recognition*, vol. 76, pp. 323–338, 2018.

[196] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual Tracking with Fully Convolutional Networks." in *International Conference on Computer Vision*. IEEE Computer Society, 2015, pp. 3119–3127.

[197] H. Nam and B. Han, "Learning Multi-domain Convolutional Neural Networks for Visual Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2016, pp. 4293–4302.

[198] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-End Representation Learning for Correlation Filter Based Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017, pp. 5000–5008.

[199] L. Zhang, A. Gonzalez-Garcia, J. van de Weijer, M. Danelljan, and F. S. Khan, "Learning the Model Update for Siamese Trackers." in *International Conference on Computer Vision*. IEEE Computer Society, 2019, pp. 4009–4018.

[200] X. Li, C. Ma, B. Wu, Z. He, and M.-H. Yang, "Target-Aware Deep Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2019, pp. 1369–1378.

[201] M. Danelljan, L. V. Gool, and R. Timofte, "Probabilistic Regression for Visual Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2020, pp. 7181–7190.

[202] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte, "Know Your Surroundings: Exploiting Scene Information for Object Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 12368. Springer, 2020, pp. 205–221.

[203] K. Yang, Z. He, Z. Zhou, and N. Fan, "SiamAtt: Siamese Attention Network for Visual Tracking." *Knowledge-Based Systems*, vol. 203, pp. 106 079–106 088, 2020.

[204] Z. Zhang, Y. Liu, X. Wang, B. Li, and W. Hu, "Learn to Match: Automatic Matching Network Design for Visual Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2021, pp. 13 319–13 328.

[205] B. Yan, H. Peng, J. Fu, D. Wang, and H. Lu, "Learning Spatio-Temporal Transformer for Visual Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2021, pp. 10 428–10 437.

[206] M. Zhao, K. Okada, and M. Inaba, "Trtr: Visual tracking with transformer." *Computing Research Repository*, vol. abs/2105.03817, 2021.

[207] Z. Wang, H. Zhao, Y.-L. Li, S. Wang, P. Torr, and L. Bertinetto, "Do Different Tracking Tasks Require Different Appearance Models?" *Advances in Neural Information Processing Systems*, vol. 34, pp. 726–738, 2021.

[208] M. Paul, M. Danelljan, C. Mayer, and L. Van Gool, "Robust Visual Tracking by Segmentation." *arXiv preprint arXiv:2203.11191*, 2022.

[209] F. Ma, M. Z. Shou, L. Zhu, H. Fan, Y. Xu, Y. Yang, and Z. Yan, "Unified Transformer Tracker for Object Tracking." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8781–8790.

[210] S. Dubuisson and C. Gonzales, "A Survey of Datasets for Visual Tracking." *Machine Vision and Applications*, vol. 27, no. 1, pp. 23–52, 2016.

[211] e. a. Kristan, Matej, "The Seventh Visual Object Tracking VOT2019 Challenge Results." in *International Conference on Computer Vision Workshops*. IEEE, 2019, pp. 2206–2241.

[212] 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.

[213] M. Mueller, N. Smith, and B. Ghanem, "A Benchmark and Simulator for UAV Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 9905. Springer, 2016, pp. 445–461.

[214] Cehovin, Luka and Leonardis, Ales and Kristan, Matej, "Visual Object Tracking Performance Measures Revisited." *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1261–1274, 2016.

[215] Erdem, Çigdem Eroglu and Sankur, Bulent and Tekalp, A. Murat, "Performance Measures for video Object Segmentation and Tracking." *IEEE Transactions on Image Processing*, vol. 13, no. 7, pp. 937–951, 2004.

[216] L. C. Zajc, A. Lukezic, A. Leonardis, and M. Kristan, "Beyond Standard Benchmarks: Parameterizing Performance Evaluation in Visual Object Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2017, pp. 3343–3351.

[217] M. Yang, C. Zhang, Y. Wu, M. Pei, and Y. Jia, "Robust Object Tracking via Online Multiple Instance Metric Learning." in *ICME Workshops*. IEEE Computer Society, 2013, pp. 1–4.

[218] S. Li and D. Y. Yeung,"Visual Object Tracking for Unmanned Aerial Vehicles: A Benchmark and New Motion Models." in *Association for the Advancement of Artificial Intelligence*. AAAI Press, 2017, pp. 4140–4146.

[219] 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*, vol. 43, no. 5, pp. 1562–1577, 2021.

[220] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2019, pp. 5374–5383.

[221] J. Valmadre, L. Bertinetto, J. F. Henriques, R. Tao, A. Vedaldi, A. W. M. Smeulders, P. H. S. Torr, and E. Gavves, "Long-Term Tracking in the Wild: A Benchmark." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 11207. Springer, 2018, pp. 692–707.

[222] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for Speed: A Benchmark for Higher Frame Rate Object Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2017, pp. 1134–1143.

[223] M. Müller, A. Bibi, S. Giancola, S. Al-Subaihi, and B. Ghanem, "TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 11205. Springer, 2018, pp. 310–327.

[224] P. Liang, E. Blasch, and H. Ling, "Encoding Color Information for Visual Tracking: Algorithms and Benchmark." *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5630–5644, 2015.

[225] J. Lim, D. A. Ross, R.-S. Lin, and M.-H. Yang, "Incremental Learning for Visual Tracking." in *Conference on Neural Information Processing Systems.*, 2004, pp. 793–800.

[226] Y. Pang, X. Shi, B. Jia, E. Blasch, C. Sheaff, K. D. Pham, G. Chen, and H. Ling, "Multiway Histogram Intersection for Multi-Target Tracking." in *FUSION*. IEEE, 2015, pp. 1938–1945.

[227] Q. Zhao, Z. Yang, and H. Tao, "Differential Earth Mover's Distance with Its Applications to Visual Tracking." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 274–287, 2010.

[228] I. Fister, I. F. Jr., X.-S. Yang, and J. Brest, "A Comprehensive Review of Firefly Algorithms." *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.

[229] 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.

[230] B. Li and L. Han, "Distance Weighted Cosine Similarity Measure for Text Classification." in *IDEAL*, ser. Lecture Notes in Computer Science, vol. 8206. Springer, 2013, pp. 611–618.

[231] N. S. Peng, J. Yang, and D. K. Zhou, "Study on Bhattacharyya Coefficients within Mean-Shift Framework and its Application." *Soft Computing*, vol. 10, no. 12, pp. 1127–1134, 2006.

[232] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "Object Tracking with an Adaptive Color-Based Particle Filter." in *DAGM-Symposium*, ser. Lecture Notes in Computer Science, vol. 2449. Springer, 2002, pp. 353–360.

[233] Q. Chen, Q.-S. Sun, P.-A. Heng, and D.-S. Xia, "Two-Stage Object Tracking Method Based on Kernel and Active Contour." *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 4, pp. 605–609, 2010.

[234] M. S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking." *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[235] C. Yang, R. Duraiswami, and L. S. Davis, "Fast Multiple Object Tracking via a Hierarchical Particle Filter." in *International Conference on Computer Vision*. IEEE Computer Society, 2005, pp. 212–219.

[236] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization." *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[237] K. Deb, A. Anand, and D. Joshi, "A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization." *Evolutionary Computation*, vol. 10, no. 4, pp. 345–369, 2002.

[238] O. Cordón García, F. Herrera Triguero, and T. Stützle, "A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and New Trends." *Mathware & Soft Computing*, 2002.

[239] E. Emary, H. M. Zawbaa, C. Grosan, and A. E. Hassenian, "Feature Subset Selection Approach by Gray-Wolf Optimization." in *International Afro-European Conference for Industrial Advancement*, ser. Advances in Intelligent Systems and Computing, vol. 334. Springer, 2014, pp. 1–13.

[240] X.-S. Yang and S. Deb, "Cuckoo Search: Recent Advances and Applications." *Neural Computing and Applications*, vol. 24, no. 1, pp. 169–174, 2014.

[241] T. Kurban, P. Çivicioglu, R. Kurban, and E. Besdok, "Comparison of Evolutionary and Swarm Based Computational Techniques for Multilevel Color Image Thresholding." *Applied Soft Computing*, vol. 23, pp. 128–143, 2014.

[242] M. Settles and T. Soule, "Breeding Swarms: A GA/PSO hybrid." in *Genetic and Evolutionary Computation Conference*. ACM, 2005, pp. 161–168.

[243] R. Thangaraj, M. Pant, A. Abraham, and P. Bouvry, "Particle Swarm Optimization: Hybridization Perspectives and Experimental Illustrations," *Applied Mathematics and Computation*, vol. 217, no. 12, pp. 5208–5226, 2011.

[244] A. Ritthipakdee, A. Thammano, N. Premasathian, and D. Jitkongchuen, "Firefly Mating Algorithm for Continuous Optimization Problems." *Computational Intelligence and Neuroscience*, vol. 2017, pp. 8 034 573:1–8 034 573:10, 2017.

[245] S. Oron, A. Bar-Hillel, and S. Avidan, "Extended Lucas-Kanade Tracking." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 8693. Springer, 2014, pp. 142–156.

[246] B. Biswas, S. K. Ghosh, M. Hore, and A. Ghosh, "SIFT-Based Visual Tracking using Optical Flow and Belief Propagation Algorithm." *The Computer Journal*, vol. 65, no. 1, pp. 1–17, 2022.

[247] P. V. Gehler and S. Nowozin, "On Feature Combination for Multiclass Object Classification." in *International Conference on Computer Vision*. IEEE Computer Society, 2009, pp. 221–228.

[248] F. S. Khan, R. M. Anwer, J. van de Weijer, A. D. Bagdanov, M. Vanrell, and A. M. López, "Color Attributes for Object Detection." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2012, pp. 3306–3313.

[249] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2005, pp. 886–893.

[250] A. Gupta and M. S. Balan, "Action Recognition from Optical Flow Visualizations," in *Conference on Computer Vision & Image Processing*, ser. Advances in Intelligent Systems and Computing, vol. 703. Springer, 2017, pp. 397–408.

[251] Y. Fang and J. Li, "A Review of Tournament Selection in Genetic Programming." in *International Symposium on Intelligence Computation and Applications*, ser. Lecture Notes in Computer Science, vol. 6382. Springer, 2010, pp. 181–192.

[252] M. E. Maresca and A. Petrosino, "Clustering Local Motion Estimates for Robust and Efficient Object Tracking." in *European Conference on Computer Vision Workshops*, ser. Lecture Notes in Computer Science, vol. 8926. Springer, 2014, pp. 244–253.

[253] S. Zhang, H. Yao, H. Zhou, X. Sun, and S. Liu, "Robust Visual Tracking Based on Online Learning Sparse Representation." *Neurocomputing*, vol. 100, pp. 31–40, 2013.

[254] F. Battistone, A. Petrosino, and V. Santopietro, "Watch Out: Embedded Video Tracking with BST for Unmanned Aerial Vehicles." *Journal of Signal Processing Systems*, vol. 90, no. 6, pp. 891–900, 2018.

[255] Q. Wang, F. Chen, W. Xu, and M.-H. Yang, "Online Discriminative Object Tracking with Local Sparse Representation." in *IEEE Winter Conference on Applications of Computer Vision*. IEEE Computer Society, 2012, pp. 425–432.

[256] X. Ren and D. Ramanan, "Histograms of Sparse Codes for Object Detection." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2013, pp. 3246–3253.

[257] S. Zhang, H. Yao, X. Sun, and X. Lu, "Sparse Coding Based Visual Tracking: Review and Experimental Comparison." *Pattern Recognition*, vol. 46, no. 7, pp. 1772–1788, 2013.

[258] M. Aharon, M. Elad, and A. M. Bruckstein, "K-SVD An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation." *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[259] Z. Jiang, Z. Lin, and L. S. Davis, "Learning a Discriminative Dictionary for Sparse Coding via Label Consistent K-SVD." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2011, pp. 1697–1704.

[260] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran, "Proximal Alternating Minimization and Projection Methods for Nonconvex Problems: An Approach Based on the Kurdyka-Lojasiewicz Inequality." *Mathematics of Operations Research*, vol. 35, no. 2, pp. 438–457, 2010.

[261] Y. Xie, W. Zhang, C. Li, S. Lin, Y. Qu, and Y. Zhang, "Discriminative Object Tracking via Sparse Representation and Online Dictionary Learning." *IEEE Transactions on Cybernetics*, vol. 44, no. 4, pp. 539–553, 2014.

[262] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A Survey of Sparse Representation: Algorithms and Applications." *IEEE Access*, vol. 3, pp. 490–530, 2015.

[263] T. T. Cai and L. Wang, "Orthogonal Matching Pursuit for Sparse Signal Recovery with Noise." *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4680–4688, 2011.

[264] J. Mairal, F. R. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative Learned Dictionaries for Local Image Analysis." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2008.

[265] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[266] D. Wang, H. Lu, and M.-H. Yang, "Online Object Tracking with Sparse Prototypes." *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 314–325, 2013.

[267] F. Yang, H. Lu, and Y.-W. Chen, "Bag of Features Tracking." in *International Conference on Pattern Recognition*. IEEE Computer Society, 2010, pp. 153–156.

[268] S. Zhang, H. Yao, X. Sun, and S. Liu, "Robust Object Tracking Based on Sparse Representation." in *IEEE International Conference on Visual Communications and Image Processing*, ser. SPIE Proceedings, vol. 7744. SPIE, 2010, p. 77441N.

[269] P. Sarkar, *Sequential Monte Carlo Methods in Practice.*, ser. Statistics for Engineering and Information Science. Springer, 2001.

[270] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust Visual Tracking via Multi-Task Sparse Learning." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2012, pp. 2042–2049.

[271] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for Sparse Representation Modeling." *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[272] J. Wang, S. Kwon, and B. Shim, "Generalized Orthogonal Matching Pursuit." *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6202–6216, 2012.

[273] I. F. Gorodnitsky and B. D. Rao, "Sparse Signal Reconstruction from Limited Data Using FOCUSS: A Re-Weighted Minimum Norm Algorithm." *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 600–616, 1997.

[274] N. Wang, J. Wang, and D.-Y. Yeung, "Online Robust Non-negative Dictionary Learning for Visual Tracking." in *International Conference on Computer Vision*. IEEE Computer Society, 2013, pp. 657–664.

[275] N. Ahmed, T. R. Natarajan, and K. R. Rao, "Discrete Cosine Transform." *IEEE Transactions on Computers*, vol. 23, no. 1, pp. 90–93, 1974.

[276] G.-P. Bonneau, "An Introduction to Wavelets for Scientific Visualization." in *Scientific Visualization*. IEEE Computer Society, 1997, pp. 16–22.

[277] J. Ma and G. Plonka, "The Curvelet Transform." *IEEE Signal Processing Magazine.*, vol. 27, no. 2, pp. 118–133, 2010.

[278] A. Lukasová, "Hierarchical Agglomerative Clustering Procedure." *Pattern Recognition*, vol. 11, no. 5-6, pp. 365–381, 1979.

[279] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[280] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient Implementation of the K-SVD Algorithm Using Batch Orthogonal Matching Pursuit." *CS Technion*, vol. 40, no. 8, pp. 1–15, 2008.

[281] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[282] Wang, Xin and Hou, Zhiqiang and Yu, Wangsheng and Pu, Lei and Jin, Zefenfen and Qin, Xianxiang, "Robust Occlusion-Aware Part-Based Visual Tracking with Object Scale Adaptation." *Pattern Recognition*, vol. 81, pp. 456–470, 2018.

[283] W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review." *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[284] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning Optical Flow with Convolutional Networks." in *International Conference on Computer Vision*. IEEE Computer Society, 2015, pp. 2758–2766.

[285] S. Meister, J. Hur, and S. Roth, "UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss." in *Association for the Advancement of Artificial Intelligence*. AAAI Press, 2018, pp. 7251–7259.

[286] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018, pp. 8934–8943.

[287] A. Painsky and G. W. Wornell, "On the Universality of the Logistic Loss Function." in *ISIT*. IEEE, 2018, pp. 936–940.

[288] B. Lucas, "An Iterative Technique of Image Registration and its Application to Stereo." in *Proc. International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.

[289] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High Accuracy Optical Flow Estimation Based on a Theory for Warping." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 3024. Springer, 2004, pp. 25–36.

[290] T. Brox and J. Malik,"Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 500–513, 2011.

[291] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large Displacement Optical Flow with Deep Matching." in *International Conference on Computer Vision*. IEEE Computer Society, 2013, pp. 1385–1392.

[292] C. Bailer, B. Taetz, and D. Stricker, "Flow Fields: Dense Correspondence Fields for Highly Accurate Large Displacement Optical Flow Estimation." in *International Conference on Computer Vision*. IEEE Computer Society, 2015, pp. 4015–4023.

[293] Q. Chen and V. Koltun,"Full Flow: Optical Flow Estimation By Global Optimization over Regular Grids." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2016, pp. 4706–4714.

[294] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2015, pp. 1164–1172.

[295] S. Park and N. Kwak, "Illumination Robust Optical Flow Illumination-Chromaticity Decoupling." in *International Conference on Image Processing*. IEEE, 2015, pp. 1910–1914.

[296] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017, pp. 1647–1655.

[297] P. Liu, M. R. Lyu, I. King, and J. Xu, "SelFlow: Self-Supervised Learning of Optical Flow." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2019, pp. 4571–4580.

[298] P. Truong, M. Danelljan, L. V. Gool, and R. Timofte, "Learning Accurate Dense Correspondences and When To Trust Them." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2021, pp. 5714–5724.

[299] X. Xiang, M. Zhai, R. Zhang, Y. Qiao, and A. El-Saddik, "Deep Optical Flow Supervised Learning with Prior Assumptions." *IEEE Access*, vol. 6, pp. 43 222–43 232, 2018.

[300] A. Ranjan and M. J. Black, "Optical Flow Estimation Using a Spatial Pyramid Network." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017, pp. 2720–2729.

[301] Z. Teed and J. Deng, "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow." in *European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 12347. Springer, 2020, pp. 402–419.

[302] Z. Tu, W. Xie, D. Zhang, R. Poppe, R. C. Veltkamp, B. Li, and J. Yuan, "A Survey of Variational and CNNBased Optical Flow Techniques." *Signal Processing: Image Communication*, vol. 72, pp. 9–24, 2019.

[303] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz,"Models Matter, So Does Training: An Empirical Study of CNNs for Optical Flow Estimation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 6, pp. 1408–1423, 2020.

[304] P. Guha, A. Mukerjee, and K. S. Venkatesh, "OCS-14 : You Can Get Occluded in Fourteen Ways." in *International Joint Conference on Artificial Intelligence*. IJCAI/AAAI, 2011, pp. 1665–1670.

[305] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition." in *ICLR*, 2015.

[306] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song, "Recent Advances and Trends in Visual Tracking: A Review." *Neurocomputing*, vol. 74, no. 18, pp. 3823–3831, 2011.

[307] A. Lukezic, T. Vojir, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative Correlation Filter with Channel and Spatial Reliability." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2017, pp. 4847–4856.

[308] Wang, Qiang and Gao, Jin and Xing, Junliang and Zhang, Mengdan and Hu, Weiming, "DCFNet: Discriminant Correlation Filters Network for Visual Tracking." *Computing Research Repository*, vol. abs/1704.04057, 2017.

[309] C. Yang, H. Lamdouar, E. Lu, A. Zisserman, and W. Xie, "Self-supervised Video Object Segmentation by Motion Grouping," in *International Conference on Computer Vision*. IEEE, 2021, pp. 7157–7168.

[310] H. Jin, P. Favaro, and S. Soatto, "Real-Time Feature Tracking and Outlier Rejection with Changes in Illumination." in *International Conference on Computer Vision*. IEEE Computer Society, 2001, pp. 684–689.

[311] G. Yu and H. Lu, "Illumination Invariant Object Tracking with Incremental Subspace Learning." in *International Conference on Image and Graphics*. IEEE Computer Society, 2009, pp. 131–136.

[312] S. Liu, G. Liu, and H. Zhou, "A Robust Parallel Object Tracking Method for Illumination Variations." *Mobile Networks and Applications*, vol. 24, no. 1, pp. 5–17, 2019.

[313] Yuan, Yue and Chu, Jun and Leng, Lu and Miao, Jun and Kim, Byung-Gyu, "A Scale-Adaptive Object-Tracking Algorithm with Occlusion Detection." *EURASIP Journal on Image and Video Processing*, vol. 2020, no. 1, p. 7, 2020.