Tracker-Independent Drift Detection and Correction Using Segmented Objects and Features

Tarek Ghoniemy

A Thesis
in the Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Electrical and Computer Engineering) at
Concordia University
Montréal, Québec, Canada

August 2017

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Tarek Ghoniemy**

Entitled: **Tracker-Independent Drift Detection and Correction Using Segmented Objects and Features**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________ Chair

Dr. Subhash Rakheja

______________________ External Examiner

Dr. Stephane Coulombe

______________________ Concordia External Examiner

Dr. Thomas G. Fevens

______________________ Examiner

Dr. Wei-Ping Zhu

______________________ Examiner

Dr. Hassan Rivaz

______________________ Supervisor

Dr. Maria A. Amer

Approved by: ______________________

Dr. Wei-Ping Zhu, Graduate Program Director

29 Sept. 2017 ______________________

Dr. Amir Asif, Dean, Faculty of Engineering and Computer Science

Abstract

Tracker-Independent Drift Detection and Correction Using Segmented Objects and Features

Tarek Ghoniemy, Ph.D.
Concordia University, 2017

Object tracking has been an active research topic in the field of video processing. However, automated object tracking, under uncontrolled environments, is still difficult to achieve and encounters various challenges that cause the tracker to drift away from the target object. To effectively handle object or environment tracking challenges, recent powerful tracking approaches are learning-based, meaning they learn object appearance changes while tracking online. The output of such trackers is, however, limited to a bounding box representation, the center of which is considered as the estimated object location. Such bounding box may not provide accurate foreground/background discrimination and may not handle highly non-rigid objects. Moreover, the bounding box may not surround the object completely, or it may not be centered around it, which affects the accuracy of the overall tracking process. Our main objective in this work is to reduce drifts of state-of-the-art tracking algorithms (trackers) using object segmentation so to produce more accurate bounding box.

To enhance the quality of state-of-the-art trackers, this work investigates two main venues: first tracker-independent drift detection and correction using object features and second, selection of best performing parameters of Graph Cut object segmentation and of support vector machines using artificial immune system. In addition, this work proposes a framework for the evaluation and ranking of different trackers using easily interpretable performance measures, in a way to account for the presence of outliers.

For tracker-independent drift detection, we use saliency features or objectness using saliency, the ratio of the salient region corresponding to the target object with respect to the estimated bounding box is used to indicate the occurrence of tracking drift with no prior information about the target model. With objectness measures, we use both

relative area and score of the detected candidate boxes according to the objectness measure to indicate the occurrenece of the tracking drift. For drift correction, we investigate the application of object segmentation on the estimated bounding box to re-locate it around the target object. Due to its ability to lead to a global near optimal solution, we use the Graph Cut object segmentation method. We modify the Graph Cut model to incorporate an automatic seed selection module based on interest points, in addition to a template mask, to automatically initialize the segmentation across frames. However, the integration of segmentation in the tracking loop has its computational burden. In addition, the segmentation quality might be affected by tracking challenges, such as motion blur and occlusion. Accordingly, object segmentation is applied only when a drift is detected. Simulation results show that the proposed approach improves the tracking quality of five recent trackers.

Researchers often use long and tedious trial and error approaches for determining the best performing parameter configuration of a video-processing algorithm, particularly with the diverse nature of video sequences. However, such configuration does not guarantee the best performance. A little research attention has been given to study the algorithm's sensitivity to its parameters. Artificial immune system is an emergent biologically motivated computing paradigm that has the ability to reach optimal or near-optimal solutions through mutation and cloning. This work proposes the use of artificial immune system for the selection of best performing parameters of two video processing algorithms: support vector machines for object tracking and Graph Cut based object segmentation.

An increasing number of trackers are being developed and when introducing a new tracker, it is important to facilitate its evaluation and ranking in relation to others, using easy to interpret performance measures. Recent studies have shown that some measures are correlated and cannot reflect the different aspects of tracking performance when used individually. In addition, they do not incorporate robust statistics to account for the presence of outliers that might lead to insignificant results. This work proposes a framework for effective scoring and ranking of different trackers by using less correlated quality metrics, coupled with a robust estimator against dispersion. In addition, a unified performance index is proposed to facilitate the evaluation process.

ACKNOWLEDGMENTS

I wish to express my sincere gratitude and indebtedness to Dr. Maria A. Amer for her supervision, guidance, and encouragement throughout this study. Her invaluable suggestions and constructive criticism during the preparation of this thesis enabled me to present the thesis in this form.

I would like also to thank my Vidpro group members and dear friends including but not limited to, Julien Valognes, Prabhakaran Ravindran, and Saeid Vosoughi for their support and encouragement during the course of this study.

This thesis is dedicated to my parents and my brothers who have supported me all the way since the beginning of my studies.

Also, this thesis is dedicated to my outstanding wife and my children, who have been a great source of motivation and inspiration.

Finally, this thesis is dedicated to all those who believe in the richness of learning.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

AIS	Artificial Immune System
AOR	Average Overlap Ratio
BB	Bounding Box
CLE	Center Location Error
CLONALG	Clonal Selection Algorithm
CoTPS	Combined Tracking Performance Score
FPS	Frame Per Seconds
FR	Failure Rate
G-Cut	Graph Cut
MAD	Median Absolute Deviation
MRF	Markov Random Field
NCLE	Normalized Center Location Error
SVM	Support Vector Machine

Chapter 1

Introduction

1.1 Motivation

In general, given the initialized location of an object in the first frame of a video sequence, object tracking is meant to estimate the state of the target object in subsequent frames. Interests in object tracking continue to increase widely with the availability of high speed computing machines, high quality video cameras, and the need for automated video analysis in many applications such as robotics, video surveillance, and traffic management. Despite the fact that much progress has been made in recent years, developing a robust tracking algorithm is still a challenging problem due to numerous uncontrolled factors that can be object-related (appearance and scale change, deformation, fast motion, and motion blur), tracking environment-related (non-stationary scenes, cluttered scenes, and illumination change), or even tracking-system-related (real-time, automation, and low resolution constraints). These challenges can negatively affect the tracking accuracy, and a drifting problem may occur in which the tracker drifts away from the target object, or false detection may be encountered [4–7].

To handle object appearance variations effectively, adaptive methods have been proposed to update the representation of a target incrementally over time. Recent powerful tracking algorithms [2, 8–13], are learning-based methods that can deal with such appearance variations. However, the output of such tracking algorithms is limited to a bounding box (BB) representation. This BB may not handle highly non-rigid objects, or may not be centered correctly around the target object, leading to a tracking drift. The investigation of suitable approaches of drift detection and correction is thus necessary for enhanced object tracking.

A common problem in video processing algorithms, such as Graph Cut (G-Cut) object segmentation or support vector machines (SVM) in object tracking, is the parameter selection that significantly affects the algorithm accuracy. Adopting optimization techniques, such as using artificial immune system (AIS), for parameter selection reduces the experimental work to spend for selecting the best parameters, reduces the bias of human intervention, and leads to optimal or near-optimal parameters that achieve better segmentation quality [14, 15].

1.2 Problem Statement

In spite of exhaustive research work, developing a robust object tracking algorithm is still a challenging task for complex and dynamic scenes, due to the drastic appearance changes caused by illumination changes, pose changes, and shape deformation. Two main problems can be highlighted; the limitation of the tracking output of learning-based object trackers to a BB, the center of which is considered as the estimated object location, and the different environmental challenges that result in tracking drift or failure. The integration of recent powerful object segmentation into object tracking to relocate the estimated output BB around the target object, when a drift is detected, may lead to better tracking.

As tracking and segmentation algorithms incorporate numerous parameters, it is important to study the influence of each parameter on the quality. Using AIS optimization techniques for the adaptive parameter selection can be a solution.

Recent benchmarks for tracking evaluation and ranking do not include robust statistical measures to account for the presence of outliers that might lead to insignificant results. It is useful to present a framework for scoring and ranking of trackers using effective quality metrics, coupled with a robust estimator against outliers. A single (unified) performance index, in addition to new performance metrics can facilitate the ranking process.

1.3 Research Objectives

The first objective of this work is to investigate the effect of integrating tracker-independent drift detection and object segmentation for drift correction, on the overall accuracy of learning-based tracking algorithms.

The second objective is to investigate the use of AIS optimization for adaptive parameter selection in the domains of object tracking and segmentation.

The third objective is to propose a framework for scoring and ranking of trackers, using known quality metrics, coupled with a robust estimator against dispersion. The investigation of new performance metrics that facilitate trackers' evaluation is to be investigated.

1.4 Summary of Contributions

The contributions of this thesis are:

1. A method for tracker-independent saliency-based drift detection where we use the saliency features of the target object inside the estimated BB to indicate the occurrence of tracking drift without prior information about the target model (Chapter 3).
2. A method for tracker-independent drift detection using edge-based objectness measure (Appendix A).
3. A method for segmentation-based drift correction where we use an automatic seeded G-Cut segmentation and propose a two-layer seed selection method based on SIFT points and foreground/background intensity relation (Chapter 3).
4. A method for adaptive selection of parameters of SVM using the AIS clonal selection-based optimization for enhanced object tracking (Chapter 4).

5. A method for optimal selection of parameters of G-Cut segmentation using the AIS clonal selection-based optimization (Chapter 5).
6. A framework for scoring and ranking of different trackers using known quality metrics, coupled with a robust estimator to account for the presence of outliers (Chapter 6).
7. New tracking evaluation measures where we propose a unified overlap-failure performance index, recovery, drift, and pure recovery-to-drift measures to facilitate trackers' evaluation and ranking (Chapter 6).
8. A framework for the selection of the best performing configuration (parameter set), and weighting all parameters according to their influence on the tracking quality. (Appendix B).

1.5 Thesis Outlines

Chapter 2 presents object tracking approaches. The principal components of an object tracking system are briefly described. Common tracking performance measures are presented. Finally, artificial immune systems are presented.

Chapter 3 introduces the proposed tracker-independent drift detection method using saliency features prior work. Automatic seed selection and segmentation for on demand drift correction is then discussed. Objective and subjective experimental results of the proposed method, applied to five state-of-the-art trackers on a publicly available data set classified into different challenging attributes, are analyzed [1].

Chapter 4 introduces the use of artificial immune system optimization for object tracking. A method for adaptive parameter selection of SVM for enhanced object tracking is proposed. The objective and subjective experimental results of the proposed approaches, applied to STRUCK tracker, are presented.

Chapter 5 introduces a method for the selection of near-optimal Graph Cut segmentation parameters using artificial immune system, and the obtained results are summarized.

[1] The author wishes to thank Prabhakaran Ravindran for his help in running simulations of JOTS method.

Chapter 6 presents the proposed object tracking scoring and ranking framework, and the corresponding ranking measures. The objective and subjective experimental results, applied to ten state-of-the-art different performing trackers on a publicly available data set, are presented.

Chapter 7 concludes the thesis and poses possible avenues for future research work.

Appendix *A* presents a proposed tracker-independent drift detection method using edge-based objectness. The objective experimental results of the proposed method applied to different trackers on a data set of various challenges show promising results.

Appendix *B* [2] presents a proposed framework for the selection, scoring, and weighting of the parameters of the tracking algorithms. The objective experimental results of the proposed framework applied to three different performing tracking algorithms are discussed.

Due to different contributions in the thesis, the symbols of each contribution are proprietary to each contributing chapter and we give a list of symbols at the start of each chapter.

[2] The author wishes to acknowledge the partial contribution of Julien Valognes to Appendix B and Chapter 6, as part of his "Concordia Undergraduate Student Research Award". Julien helped with developing the methods in sections B.5.1, B.5.2, 6.6.2, and 6.6.3 as well as in running related simulations.

Chapter 2

Background

2.1 Overview

In this chapter, object tracking approaches are introduced and categorized from appearance modeling and segmentation points of view as discussed in section 2.2.2. Components of a generic object tracking system are briefly described in section 2.2.3. Common tracking performance measures are introduced in section 2.3. Artificial immune system (AIS) algorithm, as a powerful and adaptive machine learning tool that can be investigated to enhance the accuracy of visual object tracking, is introduced in section 2.4.

2.2 Object Tracking

2.2.1 Introduction

Visual object tracking, concerned with the problem of estimating the trajectory of an object in the image plane, has many important applications. Such applications include, but not limited to automated surveillance, traffic analysis, video indexing, human computer interaction, as well as autonomous navigation. Interests in object tracking increased widely with the availability of ultra-high speed computing machines, super-high quality video cameras, and the need for automated video analysis. Automated analysis of videos is a sophisticated operation that starts by detection of object(s) of interest, then tracking the trajectory of such object(s) across frames, and ending by trajectory analysis to understand the behavior of objects and their corresponding interactions. Even though tracking is considered an important part of the above process, it is the most error prone component.

Difficulties in object tracking arise from a variety of uncontrolled factors in the tracking environment that probably appear in the form of information loss. Loss of information may be due to scene projection, noise effect, complex object motion, camera motion, deformable object shapes, mutable object appearance, illumination changes, occlusions, and real-time constraints which impose extra level of difficulty on the tracking systems. While recent researches have introduced a significant progress in the domain-specific visual tracking, developing tracking systems that can benefit the cognitive abilities of human beings is still a challenging research problem. Typically, most existing tracking systems impose various constraints in order to simplify the tracking problem and hence, such tracking systems cannot adaptively fit in various environments. Accurate tracking requires effective modeling and representation of the tracking environment.

Numerous approaches for object tracking have been proposed. They primarily differ from each other based on the way they approach the following questions: Which object representation is suitable for tracking?, Which image features should be used?, and How should the motion, appearance, and shape of the object be modeled? Answers to these questions depend on the context/environment in which the tracking is performed and the end use for which the tracking information is being sought. Several tracking methods, that attempt to answer these questions for a variety of scenarios, have been published [4, 5, 16].

2.2.2 Classification of Object Tracking Approaches

Visual object tracking algorithms can be categorized according to different points of view. Recent studies adopt the classification of object tracking approaches into fixed and adaptive appearance modeling based methods [6]. Fixed appearance modeling based tracking methods can be sub-categorized into non-segmentation-based and segmentation-based methods. Learning-based tracking methods [6, 8–11, 13, 17, 18] do not incorporate segmentation in their tracking framework and hence, can be categorized as non-segmentation-based methods.

2.2.2.1 Fixed Appearance Modeling Based Tracking

Tracking methods with fixed models of a target prior to the start of tracking task use different methods to represent the appearance of objects such as templates and density-based approaches [4]. Template matching is the most commonly used approach in the case of single object tracking due to its simplicity. Templates, however, only encode the object appearance generated from a single view. Parametric density has been used for object representation in many tracking algorithms. Using probability density, object appearance can be estimated either parametrically, such as mixture of Gaussian (MoG), or non-parametrically, such as histograms. Mean-shift tracking approach uses a mixture of both spatial information and color histogram for object representation [19]. An obvious advantage of the Mean-shift tracker over the template matching is the elimination of an exhaustive search and accordingly, it has a good contribution for real-time applications. However, such tracking methods may fail as a result of the inevitable appearance variations that can be from the object itself such as non-rigid structure, shape deformation, posture changes and abrupt motion, or from the surrounding environment such as illumination variation, camera motion, camera scale and occlusion [6].

In general, object tracking algorithms start by detection of object of interest and then, finding the object correspondence across frames. In [4], point tracking algorithms that use probabilistic approach [20,21] to solve the correspondence problem, represent objects as points, and the association of such points is based on the previous object state which can include object position and motion. These approaches do not include a segmentation step in the tracking algorithm itself and can be categorized as segmentation-free methods. In this work, we will categorize tracking algorithms that do not incorporate a segmentation step in the tracking loop as non-segmentation-based tracking [22]. The accuracy of such approaches is coupled with the assumptions and constraints followed by the tracking algorithm.

While object segmentation is meant to partition image pixels into meaningful regions based on certain characteristics such as color or texture in a spatial domain, object tracking aims to partition such pixels based on consistence properties in a temporal domain and hence, the two tasks facilitate each other and are found to be closely related and both can be greatly improved if they are solved jointly. Accordingly, a new class of object tracking approaches, that combines tracking and segmentation in an integrated framework, is found to improve the performance of tracking systems. While solving the segmentation problem helps precluding the tracking failures, tracking at the same time provides an important input that can guide segmentation and enhance its performance. Integration of segmentation and tracking approaches is found to enhance the target localization performance, leading to a reduced tracking drift [23–27]. Such approaches are also found to use probabilistic methods or kernel based tracking methods integrated with a proper segmentation technique. However, many of such approaches impose few assumptions about object contours that must be given in the first frame. Such methods focus on explicitly integrating segmentation methods, such as graph-cuts and active contours, into object tracking in each frame to enhance the tracking accuracy [28–30]. In such approaches, segmentation algorithms are used to support the tracking rather than separating the object accurately from its surrounding. These approaches will be referred to as segmentation-based tracking approaches. These object tracking approaches apply fixed object models, and are more likely to fail as a result of inevitable appearance changes.

2.2.2.2 Adaptive Appearance Modeling (Learning) Based Tracking

To handle the appearance variations effectively, adaptive methods have been proposed to update (learn) the representation of a target incrementally over time. Recent tracking algorithms [6, 8–11, 13, 17, 18], are learning-based methods that can deal with such appearance variations, thus achieving more accurate tracking compared with fixed model-based ones. An appearance model is used to represent the object of interest (target) while the motion model predicts the likely states of target over time. In general, a learning-based object tracking system interconnects four main modules: object initialization, appearance modeling, motion estimation, and object localization as illustrated in Figure 2.1, where F_1 represents the first frame in a given video sequence and N represents the number of frames. The tracking process starts by object initialization that can be manual (user annotates object location using BB) or automatic through detection mechanism. Once the object is initialized, several factors need to be considered for a robust appearance modeling. First, the object of interest has to be efficiently represented, which concentrates on how to robustly describe the spatio-temporal characteristics of object appearance. Visual object representation can be either local (encodes local statistical information such as interest points) or global (reflects the global statistical characteristics such as color histogram). For robust tracking, adaptive methods have been proposed to update the representation of a target using statistical learning techniques.

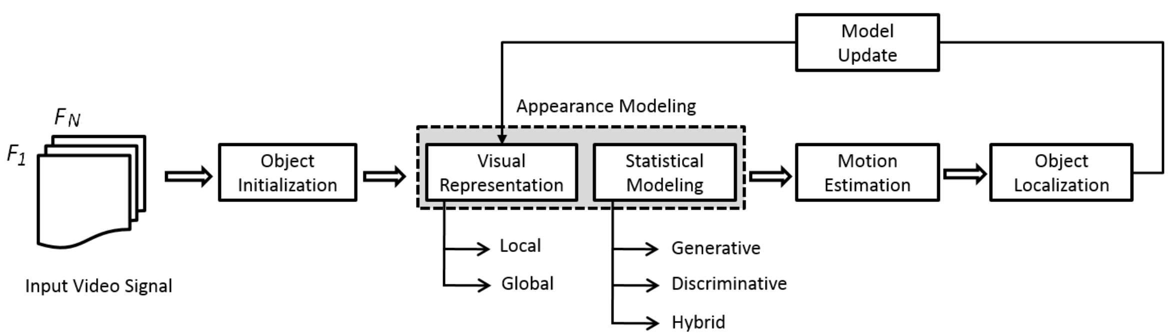


Figure 2.1: Learning-based object tracking system.

From point of view of statistical appearance modeling, recent tracking algorithms use an updating scheme to update the target model and hence, referred to as learning-based tracking algorithms. Such schemes can be generative, discriminative, or hybrid methods. For generative methods [8], tracking is formulated as searching for the region of the highest similarity with the object in neighborhood. For discriminative methods [17, 31], tracking is formulated as a classification problem that aims to discriminate the object of interest from its background. Discriminative classifiers often outperform generative models given enough training data, while generative methods often have better generalization for small size of training data. Discriminative learning is also refereed to as Tracking-By-Detection. Recently, hybrid discriminative generative methods have opened a promising direction to benefit from both types of methods [9].

After appearance modeling, motion estimation is formulated as a dynamic state estimation problem. The task of motion estimation is usually completed by utilizing a prediction module using Kalman or particle filtering [4, 32]. Kalman filter simply finds the exact solution, given a simple model under assumption that the state space model is linear and the noise follows the statistical Gaussian distribution. Such limitations can be overcome by using Particle filters. Finally, a greedy search based on motion estimation can be used for object localization and the target model is then updated.

2.2.3 Object Tracking System Components

Object tracking is a sophisticated process concerned with the estimation of target(s) trajectory. An object tracking system comprises several complementary interconnected modules as illustrated in Figure 2.2. It starts by initializing the object(s) in the first frame F_1 in the form of a bounding box B_1. Target modeling is then adopted for object state estimation through prediction. The target tracking-by-learning (learning module) is then employed to update the adaptive model over time in order to discriminate the object of interest at each frame F_t in the form of B_t. During tracking, the object detection and feature extraction modules provide the required information to improve the tracking process.

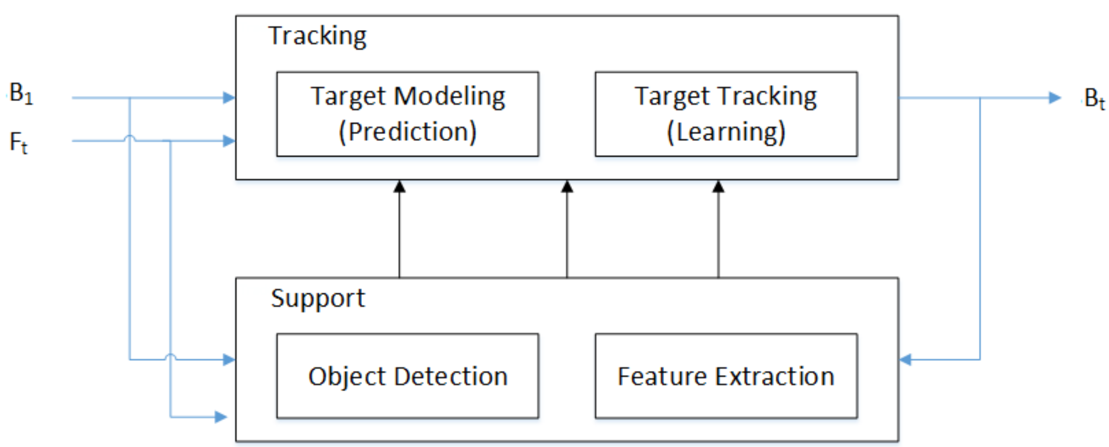


Figure 2.2: Object tracking system components.

2.2.3.1 Target Modeling (Prediction) Module

In addition to observed measurements, other information can contribute to the target state estimation. Some information may result from motion constraints of the moving object and its interaction with the environment [13]. Motion estimation is formulated as a dynamic state estimation problem and is usually adopted by utilizing a prediction module using Kalman or particle filtering [33, 34]. Kalman filter simply finds an exact solution, under assumption that the state space model is linear and the noise is statistically Gaussian. On the contrary, particle filtering can deal with nonlinear models and different forms of noise. Particle filtering simulates the state space of the system using certain number of random particles, each of which is weighted through approximation of the probability density function (PDF).

2.2.3.2 Target Tracking Module

Given the object regions in the image, it is then the trackers task to perform object correspondence across frames to generate the corresponding trajectories. Recently, object tracking is posed as a learning-based problem, where adaptive appearance models are adopted for target modeling. Such learning-based tracking can handle drastic appearance changes caused by illumination change, camera motion, pose change, and object shape deformation. In learning-based approaches, the tracking is posed as a classification problem to discriminate between the object and its surrounding. The learning strategy is embedded in the tracking framework to update the target appearance model adaptively in response to appearance variations. The essential phase of the learning module is the update phase, in which the close neighborhood of the current estimated object location is used to sample positive training examples, distant surrounding of such location is used to sample negative examples, and both are used to update the classifier over time during tracking.

2.2.3.3 Detection Module

The object detection mechanism, needed by any tracking system, is of utmost importance and can affect the performance of tracking results, especially for objects that employ a small motion across frames. A common approach for object detection is to make use of the temporal information computed across frames to detect the change in object location relative to its surroundings. Such temporal information is usually in the form of frame differencing, which highlights changing regions in consecutive frames. Object detection can be performed in a variety of ways. The most commonly known approaches for object detection are Interest points, Background modeling, and object segmentation [4].

Common detectors follow the sliding window paradigm [35, 36]. A classifier is first trained to distinguish windows containing instances of a given class from all other windows. The classifier is then used to score every window in a test image. Local maxima of the score localize instances of the class. However, this approach is class specific, and is not appropriate for automated applications such as object tracking that track different types of objects. In addition, it is computationally intensive. Objectness measures attempt to generate a small set (few hundreds or thousands) of object regions that cover every object in the input image, regardless of the specific categories of those objects (generic over classes). Compared with traditional sliding window approach, estimating object proposals in a pre-processing stage has the following advantages: 1) better accords with our human visual system behavior which perceives objects before identifying them; 2) speeds up the computation by reducing the search locations, especially when the number of object classes that need to be detected is high [37].

Recently, objectness measures [38] and saliency models [39] have occupied major research areas in object detection. Objectness approaches are related to several research strands such as interest point detectors (IPS) and saliency models (class-specific and class-generic). IPS respond to local textured image neighborhoods, and focus on individual points instead of the entire object(s) in the image scene [40]. Class-specific saliency models define, as a salient region, the visual characteristics that best distinguish a specific object class, such as vehicle or human, from others [41]. Class-generic saliency models [42–45] measure the saliency of pixels as the degree of uniqueness of their neighborhood relative to the surrounding region.

2.2.3.4 Feature Extraction Module

Feature extraction and description is an essential step in the tracking pipeline and allows us to highlight information of interest to represent a target. Extracted features can be grouped into three main classes that are low-level (color and motion), mid-level (edges, interest points, and regions), and high-level (object models) [4]. The most widely used features for object description include color, edge, optical flow, and texture. Color is one of the most widely used features for tracking, however, color spaces are sensitive to noise and illumination changes [46]. Edges are commonly used as a representative feature for applications of boundary tracking [47]. An important property of edges is that they are less sensitive to illumination changes compared to color features. Corner is closely related to algorithms that use edge analysis to find rapid changes in direction [48].

The terms corners and interest points are used interchangeably and refer to point-like features in an image, which have a local two dimensional structure. Blob provides a complementary description of image structures in terms of silhouettes [49]. Blob may sometimes also be regarded as interest point descriptors as they often contain point structures. However, Blob detectors can detect too smoothed image areas that cannot be detected by a corner detector. Optical flow is commonly used as a feature in motion-based segmentation and tracking applications [50]. Optical flow is a dense field of displacement vectors that defines the translation of each pixel in a region under constant intensity assumption constraint. Texture is a measure of the intensity variation of a surface which quantifies properties such as smoothness and regularity. Compared to color and edges, texture requires a processing step to generate the descriptors [49]. Interest-point detectors such as Scale-invariant Feature Transform (SIFT) [51], speeded up robust features (SURF) [52], oriented FAST and rotated BRIEF (ORB) [53] aim at selecting highly distinctive local image features that can be accurately localized across multiple image frames under pose and illumination variation.

2.3 Common Tracking Performance Evaluation Measures

Several performance metrics of the tracking algorithms use empirical discrepancy methods [54] that compare off-line ground-truth data with the estimated trajectories. Among such metrics, the average overlap ratio (accuracy), center location error, normalized center location error, failure rate (robustness), or derivatives thereof, such as success and precision plots are commonly used [55].

- The average overlap ratio (AOR) measures the overlap ratio between the estimated BB predicted from the tracker (B_t) and the annotated BB (B_t^g) according to $AOR = \dfrac{B_t \cap B_t^g}{B_t \cup B_t^g}$.
- The center location error (CLE) is a widely used metric that computes the average Euclidean distance between the centers $\hat{c}_e$ and $\hat{c}_g$ of the estimated B_t and the annotated B_t^g. However, when the tracker loses the target, the output location might be random and thus the measure does not reflect the actual tracking quality.

  The normalized center location error ($NCLE$) computes the normalized Euclidean distance between the centers $\hat{c}_e$ and $\hat{c}_g$ of the estimated B_t and the annotated B_t^g with respect to the ground truth BB dimensions.
- The failure rate (FR) is the percentage of the number of failures per sequence based on the overlap between the B_t and B_t^g according to $FR = \dfrac{N_Z}{N_F}$ where N_Z is the number of frames where $AOR = 0$ and N_F is the total number of frames per sequence.
- The success plot is a widely used metric for the evaluation of different tracking algorithms [56, 57]. It represents the percentage of frames for which the overlap measure exceeds a certain threshold, with respect to different thresholds.
- The precision plot is commonly used to measure the percentage of frames in which the estimated locations are within a certain threshold distance of the ground-truth

positions. Such plot is measured with respect to different thresholds in a specific range.

2.4 Artificial Immune System

Artificial Immune System (AIS) is an emergent biologically motivated computing paradigm. Its main concept is the extraction of principles from the natural immune system (NIS) in order to design alternative computational tools for complex problem solving. The main role of the immune system is to recognize and discriminate an organism from foreign elements. The capability to recognize and eliminate specific (non-self) patterns serves as a good source of inspiration to develop novel computational mechanisms for machine learning and pattern recognition [58, 59]. AIS algorithms are considered highly robust, adaptive, self-organized, and inherently parallel structured [60]. They have the ability to escape the local optimum region through mutation, and strong local search capability through cloning. They also add diversification by replacing the worst performing individuals in the population. Several AIS techniques have been developed for optimization and machine learning problems, each of which mimics a certain principal in the NIS. Among such techniques, the clonal selection algorithm (CLONALG) [59] is a widely employed AIS approach. In optimization problems, CLONALG learns to recognize patterns through an evolutionary-like procedure and is capable of solving complex engineering tasks, such as multi-modal and combinatorial optimization [59]. CLONALG algorithm is used in diverse applications including image classification [15, 61] and segmentation [62, 63]. Affinity proportional reproduction and mutation are two important features of the CLONALG algorithm.

In AIS terminology, the optimization problem to be solved is the antigene, generated solutions are the antibodies, fitness value (objective function evaluation) is the affinity, cloning is the reproduction of solutions, mutation is the random modification of solutions, and receptor editing is the diversification of solutions. For CLONALG-based AIS [1], a population of antibodies (solutions) ab is randomly generated of certain size P_S as

$$P_S = n \cdot N_d = P_M + P_R, \tag{2.1}$$

where P_S is the size of the population, n is a small number, N_d is the total number of design variables, P_M is the number of best candidate solutions, and P_R is the number of candidate solutions to be replaced by randomly created solutions. Each antibody represents a combination of alternatives of all design variables in the form of chromosomal representation. The antibodies are sorted according to their affinity into either non-dominated (superior among all antibodies) or dominated. The dominance is checked and the best P_M antibodies are selected which go through a cloning process that forms the local search tool of the algorithm. The number of clones N_{cln} to all the antibodies is selected as

$$N_{cln} = n_c \cdot P_S, \tag{2.2}$$

where n_c is a small number. The antibodies with the highest affinity are subjected to higher clones, so they are more likely to be selected as the best solutions in the next generations. Then a subset of the cloned antibodies undergoes hyper-mutation and diversification operations that form the basis of the global search mechanism of the algorithm. To perform the mutation process, the antibodies are encoded into binary strings and the mutation rate is kept inversely proportional to antigene affinity. A percentage of the worst members of the previous population of antibodies is replaced with some randomly generated new solutions which will add diversity to the population. The cloning and mutation processes increase the tendency to achieve the optimal solution. The memory (archive) of size A_S is utilized to store the best candidate antibodies among generations and is defined as

$$A_S > div^k - (div - 1)^k + 2k, \tag{2.3}$$

where $div > 2k$, is the number of divisions used to identify the crowdedness of the solutions and k is the number of objectives in the optimizatiion problem. When copying the best antibodies to the archive, if the archive is not full, all the non-dominated antibodies are allowed to enter the archive. If the archive is full, the best antibodies which belong to the lowest crowded region are allowed to enter the archive, and spontaneously make random elimination of the antibodies which belong to the most crowded regions with the same rate of the newly introduced antibodies. The length l_e of the encoded binary string for each antibody is calculated as in

$$2^{l_e} > U_b, \tag{2.4}$$

where U_b is the upper boundary of the corresponding design variable.

Figure 2.3 illustrates the detailed flowchart of the clonal selection algorithm, which randomly generates P_S solutions of the optimization problem. The best antibodies according to a pre-defined size go through cloning and mutation process to construct new candidate solutions. These solutions are evaluated and a percentage of the best solutions P_M is added to the population. Further, a percentage of worst P_R antibodies are discarded and replaced with new randomly created solutions. Note that symbols in Figure 2.3 represent the number of solutions. We aim to show that the number of solutions is the same and the candidate solutions may be changed to find the near optimal ones. In the flowchart, we add the non-best solutions, fraction of the best solutions (where we applied clonning and hyper-mutations), and fraction of the rest of the best solutions to form the solutions in the current iteration.

The CLONALG undergoes four steps to reach the final near optimal solution [64]:

- generation of random population which is a pool of antibodies or immune cells,
- proliferation of best antibodies which is simply performed through cloning process,
- hyper-mutation of clones (blind variation) to maintain diversity by applying random genetic changes, and
- affinity of antigene antibody interaction through the evaluation of the objective function and elimination of low affinity antibodies.

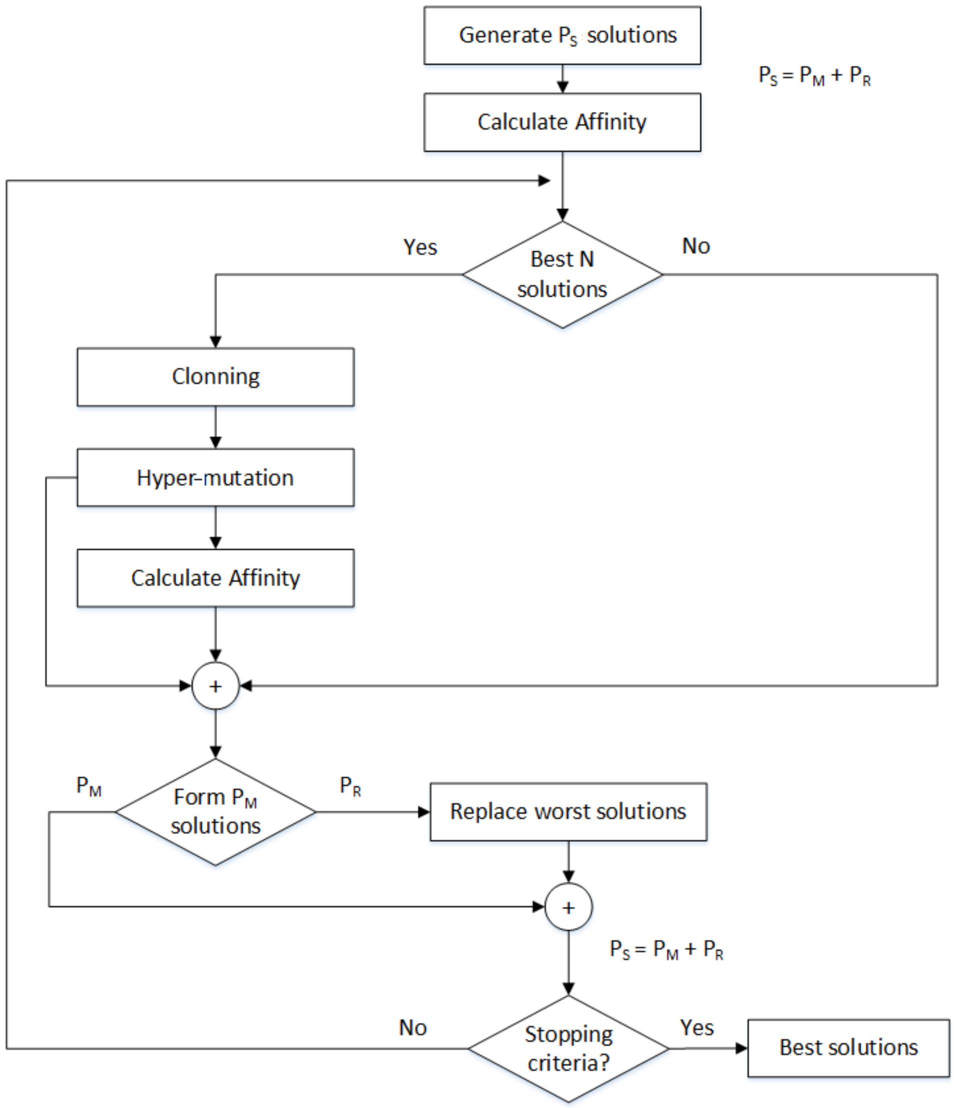


Figure 2.3: Clonal selection algorithm.

Consequently, the best antibody or group of design variables which achieve the best objective function value will continue for more processes in the algorithm and the rest with low affinity will be removed. The CLONALG possesses the following three techniques to maintain diversity that improves the ability to find a solution closer or at the global optimal, preventing from stuck into local minima [1, 65]:

- hyper-mutation,
- receptor editing, which is called non-uniform mutation, and
- a fraction of new antibodies are added to the generated solutions.

The diversity maintained by the non-uniform mutation helps the antibody-antigene affinity to escape from local minima in the affinity landscape as shown in Figure 2.4. As illustrated, the uniform mutation allows an antibody A to search small local searches of antibodies with higher affinities (A^1), because low affinities are eliminated, while non-uniform mutation allows large search area steps, where the affinity might be lower (an antibody A to an antibody B) or higher (A to C), in which mutation will lead to reach a solution near to the global optimum.

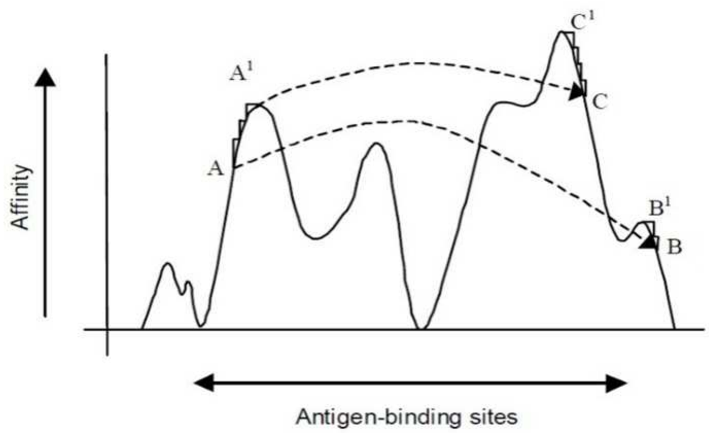


Figure 2.4: Non-Uniform mutation process of antibodies [1].

2.5 Summary

In this chapter, object tracking approaches are presented, with a focus on the adaptive appearance modeling-based approaches. The principal components of object tracking systems are introduced. Common performance measures are then introduced. The CLONALG artificial immune system algorithm is then presented.

Chapter 3

Tracker-independent Drift Detection and Correction

3.1 Abstract

Accurate object tracking is still a challenging problem due to numerous factors, that may cause the tracker to drift away from the target object. Some trackers use segmentation to enhance the tracking quality. Recent learning-based trackers perform much better than segmentation-based ones. However, their output is a bounding box that may not well discriminate foreground and background and may not be centered correctly around the target object. This chapter proposes a method that detects drift of a tracker, using saliency features of the target objects. If the tracker tends to drift or shows inaccuracies, we propose a method that applies automatic seeded object segmentation on the estimated tracking output to correct the drift. Such segmentation is meant to re-locate the bounding box around the target object. As seeds for segmentation, we propose to use SIFT interest points conditioned they are non-background pixels. Results on a publicly available benchmark of 100 sequences that cover various tracking challenges show the ability of the proposed method to improve the tracking quality of five recent, and different performing, trackers. Simulation also show that the proposed method outperforms segmentation-based trackers.

3.2 List of Symbols

Symbol	Description
W	Width of bounding box
H	Height of bounding box
V	Half of minimum dimension of object bounding box
M	Segmentation mask at frame F_t
C	Target object contour at frame F_t
S	Seed Mask of the estimated bounding box
B_t	Estimated bounding box at frame F_t
B_t^r	Relocated bounding box at frame F_t
B_t^g	Ground-truth bounding box at frame F_t
d	Distance from the bounding box boundary toward center
μ_o	Average intensity of object
μ_b	Average intensity of background
s_k	Saliency map of the target at frame F_t
$\{b_l\}$	Binary mask of saliency map at frame F_t
α_s	Ratio of binary salient pixels inside saliecy map
n_i	Number of pixels in region r_i
$D_{lab}(r_k,r_i)$	Color distance in LAB color space between region r_k and region r_i
$E(r_k,r_i)$	Spatial distance between centers of region r_k and region r_i
σ_s	Term to control the strength of spatial weighting
N_b	Number of binary pixels of the salient object
p_i	Interest point number i

3.3 Introduction

Object tracking is a demanding application. Given the initial location of a target in the first frame, it estimates the states of the target in subsequent frames. Despite the fact that much progress has been made in recent years, developing a robust (no-drift) tracking algorithm is still a challenging problem due to numerous uncontrolled factors. Such factors can be object-related (appearance and scale change, deformation, fast motion, motion blur, or occlusion), environment-related (non-stationary scenes, cluttered background, or illumination changes), system-related (real-time and automation constraints), or combinations thereof [4]. The above-mentioned factors may cause the tracker to drift away from the target object [4]. Drift detection is crucial, as it allows the tracking algorithm to start a recovery (drift correction) process in order to maximize the tracking accuracy. Drift detection can be based either on prior information about the target object and tracking environment [66, 67], or on features of the target object such as visual saliency.

This chapter first proposes a method for drift detection using saliency features. Visual saliency is the perceptual quality that makes an object stands out relative to its surrounding and thus captures attention. Detection of salient regions of an image has diverse applications, including object detection and segmentation [68–70], recognition [71], and image retrieval [72]. Saliency is also used as cue to measure how likely an image window contains an object [38]. Using saliency features for drift detection has several advantages. First, no prior information is needed. Second, the saliency detection process is not computationally expensive. Finally, one can still get the saliency information even under challenging conditions such as occlusion and illumination variations [73].

Object segmentation can be used to improve the accuracy of object tracking [74]. Segmentation-based tracking approaches provide segmentation input to the tracking algorithm in a closed loop form, for successful tracking [23–27]. However, such approaches are not competitive in accuracy with learning-based tracking approaches such as [2, 10–13]. On the other side, the output of learning-based trackers is, limited to a bounding box (BB). Such BB may not accurately discriminate foreground and background, handle non-rigid objects, or be centered accurately around the target object, which affects the accuracy of the overall tracking process. This chapter thus proposes drift correction by applying automatic seeded object segmentation on the tracker's output BB for enhanced tracking quality through drift reduction. Applying object segmentation on each output BB of the tracker is assumed to provide more accurate BB location with respect to the target object. However, running the segmentation each frame has two main drawbacks: first, it is computationally intensive; second, segmentation result may become inaccurate under video challenges such as motion blur or occlusion. Accordingly, we propose to apply object segmentation only when a tracking drift is detected.

In the rest of the chapter, section 3.4 presents prior work, and its relation to the proposed approach, which is introduced in section 3.5; section 3.6 presents the analysis and discussion of the obtained results; and section 3.7 concludes the work and proposes future work.

3.4 Prior Work

Various approaches have been used for enhancing object tracking quality such as integrating segmentation and tracking in a closed loop as well as template matching. Object tracking approaches can be divided into those explicitly using object segmentation [23–27], and those not making explicit use of it, such as learning-based tracking methods [2,10–13]. In the following, we review both categories and also methods that explicitly handle drift detection and correction.

3.4.1 Segmentation-based Trackers

Segmentation-based tracking approaches use segmentation to initialize the tracking per frame in a closed loop form. In [23], a fine Random-Walker (R-Walk) segmentation of an object at any frame is used to initialize the tracking for the next frame. In [24], a closed loop interaction between EM-like color-histogram tracking and R-Walk segmentation has enhanced the accuracy of object localization. The spatial properties and appearance of segmented objects are exploited to initialize the tracking algorithm in the next step. In [25], G-Cut segmentation is applied to mean-shift tracking in a closed loop. Integrating G-Cut within the optical flow tracker in [26] showed the ability to track articulated objects under challenging conditions. These methods require user input at the first frame.

In [74], Wen et al. presented a joint tracking and segmentation (JOTS) algorithm which integrates multi-part tracking and segmentation into a unified energy optimization framework. The multi-part tracking and segmentation are carried out iteratively to minimize an objective function using a RANSAC-style approach. JOTS uses the SLIC super pixel for multi-part segmentation and the segmentation is used to initialize the tracking at next incoming frame. In such approaches, the segmentation is crucial to initialize the tracking algorithm at each frame for successful tracking. However, running segmentation at each frame has two main drawbacks: first, it is computationally intensive; second, segmentation may become inaccurate under video challenges such as motion blur which may mislead the tracking process.

In the above methods, seeds through interactive user input are required. Interactive seeded segmentation has appealing results [75–78] which require input seeds to represent both object and background through user interaction. However, it is impractical for automated tracking applications. [79] presented a method of object recognition and segmentation using Scale-Invariant Feature Transform (SIFT) and G-Cut. However, this method assumes that the object models, used for filtering of the interest points, are pre-selected, which is not always available. In [80], J. Shan et al., presented a new automatic seed point selecting method for region growing algorithm for breast lesion's images. One of the limitations of this method is that results are affected by shadow with similar intensity of the lesion region, which is a case that can commonly occur in tracking environment. [81] used the initial contour (not seed points) close to the object boundary to initialize the active contour segmentation. The initial contour of the level set segmentation is a closed curve. Therefore, the convex-hull polygon is chosen to embody the salient object points. However, the active contour segmentation is computationally intensive. In [82], Yang et al. presented an automatic color image segmentation using G-Cut and color SIFT (CSIFT) features. They assume that pre-captured models of the colored target object are available.

3.4.2 Learning-based Trackers

The last few years have witnessed the emergence of several learning-based high performing trackers [2, 3, 8–11, 13, 18, 83–85]. STRUCK tracker [2] is an adaptive tracker based on kernelized structured output prediction using support vector machine, which is learned incrementally over time. Sequential minimal optimization (SMO) is adopted to find the optimal support vectors from samples around estimated object location to update the classifier for target prediction. ASLA tracker [8] uses a structural local sparse appearance model that exploits both partial and spatial information from sampled candidate patches around estimated object location. A dictionary learning based on a structured sparse representation is combined with robust sparse coding in which the learned classifier is employed to separate the object from background. SCM tracker [9] uses a collaborative model with an updating scheme that considers both the latest observations and the original template, thereby handles appearance changes effectively. KCF tracker [10] is a Kernelized Correlation Filter operating on simple HOG features that performs training and detection to discriminate an object appearance from its surrounding. SAMF tracker [11] is a correlation filter (CF) based tracker that uses a scale adaptive scheme to tackle the problem of the fixed template size in the CF. LOT tracker [12] automatically estimates and adapts, on-line, to the rigidity of the tracked object through a probabilistic model to handle appearance variations over time. DSST tracker [13] extends the Minimum Output Sum of Squared Errors tracker [86] with robust scale estimation. In addition, DSST learns a one-dimensional discriminative scale filter to estimate target size. STAPLE tracker [3] combines two image patch representations to learn a model that is inherently robust to both color changes and deformations. Two independent ridge regression problems are solved, exploiting the inherent structure of each representation to maintain real-time performance. STAPLE combines the scores of template and histogram models in a dense translation search, that are learned independently, enabling greater accuracy. CCOT [84] introduced a formulation for training continuous convolution filters. It employs an implicit interpolation model to pose the learning problem in the continuous spatial domain which enables the efficient integration of multi-resolution deep feature maps. T-CNN [83] presented an online visual tracking algorithm by managing multiple appearance models in a tree structure. Such algorithm employs Convolutional Neural Networks (CNNs) to represent target appearances. It is convenient to handle multi-modality in appearances and preserve model reliability through smooth updates along tree paths.

Deep learning is one of the most successful research directions in machine learning and computer vision. In object tracking, it detects candidate targets in consecutive frames in which deep learning is used to recognize the object of interest among such candidates. The power of deep learning appears in its ability for automatic feature expression. Through a multi-layered learning architecture, the deep networks can achieve both high dimensional and abstraction level with obvious distinction. One of the most popular deep-learning architectures is the convolution neural network (CNN) [87]. Due to its superiority among other architectures, CNN becomes the mainstream model in visual tracking. Generally, an off-line trained large-scale CNN is adopted for both classification and tracking. The common approaches of CNN-based tracking are both fully CNN (FCNT) [85] and multi-domain CNN (MD-Net) [18]. FCNT constructs a feature selection network in addition to prediction networks. In addition, such networks are found to use irreverent image data to reduce the training demand, which causes deviation from tracking to some extent [87]. In the pre-training phase, the object of certain class in one video can be a background in another video. As a result MD-Net defines a domain to be a set of videos that contain same kind of objects and proposes to use a multi-domain structure to distinguish between the object and background in each domain independently. Deep learning approaches have some limitations. First, the pre-training is inefficient for on-line training that affects the tracking performance. Second, it incorporates a large number of parameters that are not shared among different layers. Third, MD-Net does not fully utilize video information in temporal domain. Recent network models, such as recurrent neural networks, are being discovered and showed to outperform the FCNT and MD-Net.

Learning-based trackers output a BB that is not adaptive to object boundaries and shape, specially when the target object undergoes drastic appearance changes. Also once the tracker starts drifting, the location and overlap error accumulate quickly, distorting object model recursively and eventually leading up to a total failure.

3.4.3 Trackers with Drift Detection and Correction

Drift detection allows the tracking algorithm to start a drift correction (recovery) process. To detect drift from the target object, object detection methods can be used. State-of-the-art object detectors follow the sliding window paradigm [35,36]; they classify first windows containing instances of a given class. The classifier is then used to score every window in a test image in which a local maximum of the score localizes instances of such class. After drift detection, a recovery process is important to correct the drift. In [67], Schreiber proposed a modified Lucas-Kanade template matching with drift correction, in which an object is tracked by extracting a template in the first frame and then finding the region which matches the template as closely as possible in the remaining frames. The underlying assumption is that the object appearance remains the same. In [66] the current estimated template is updated using naive algorithm and then, aligned with the retained first frame template to give the final update. Such drift correcting algorithm is still sensitive to variations in the object appearance relative to the first template.

3.4.4 Differences to Our Approach

Our proposed method 1) is tracker independent, and can be applied to any tracking algorithm; 2) requires no prior information about the target object for automatic drift detection; 3) uses automatic segmentation with robust seed selection through both SIFT and intensity features to filter out seeds related to the background regions; and 4) outputs a BB more adaptive to object boundaries and shape. Our method differs from segmentation-based tracking, as it applies segmentation only when a drift is detected and hence, it can achieve both better quality as well as higher frame rate.

3.5 Proposed Method

The proposed method comprises two main components as shown in Figure 3.1: drift detection using saliency features and drift correction using seeded segmentation. At current frame F_t, given the estimated tracking output BB from previous frame F_{t-1}, an object tracker estimates the BB (B_t) around the target. If a drift is detected, the drift correction relocates the BB around the target through automatic seeded segmentation.

3.5.1 Saliency-based Drift Detection

Saliency object detection, sometimes called salient segmentation, is interpreted in computer vision as the process that incorporates detection of the most salient region(s)

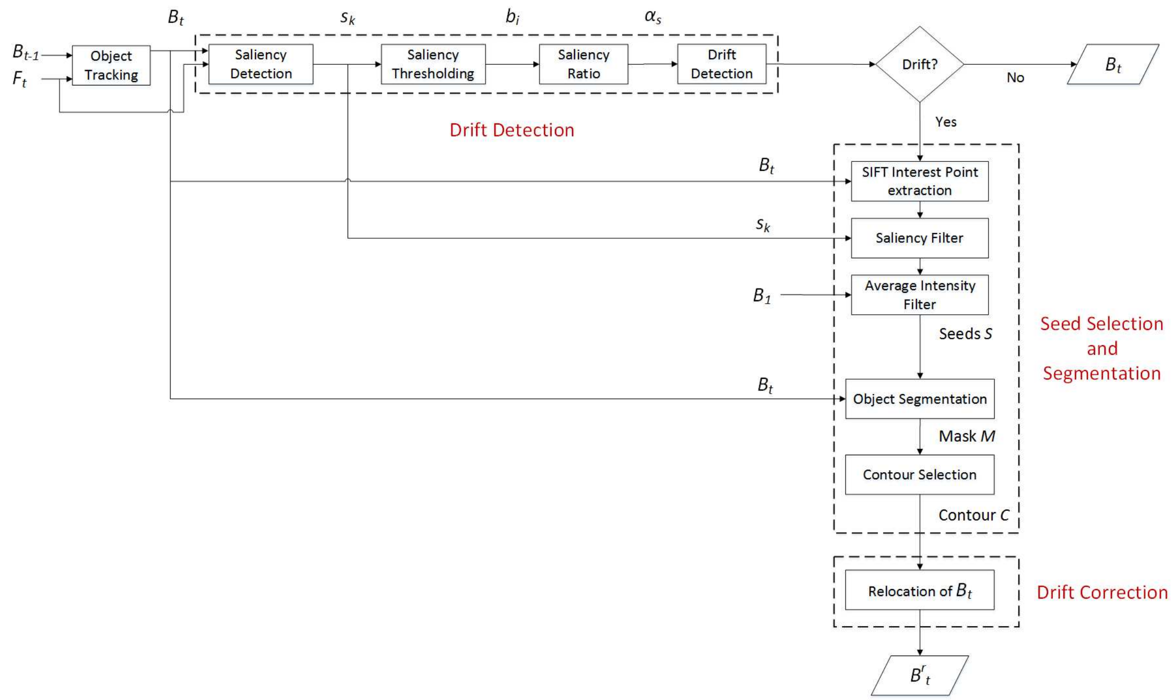


Figure 3.1: Block diagram of the proposed drift detection and correction method.

in an image and then segmenting the boundary of such region(s). For saliency of a region, a high contrast to its surrounding regions is usually stronger evidence than that of far-away regions. Generally, an object is more likely to be salient than a region on the background, as image background is usually more structured and homogeneous (thus less salient) than objects [88].

While most of saliency models [42, 43, 45] employ local contrast, we calculate the saliency map more robustly [44] using global contrast differences and spatial coherence. However, directly introducing the spatial relation among individual pixels is computationally expensive and thus, we partition the BB B_t into K regions (e.g., using [89]) and calculate the saliency s_k of each region r_k as a weighted sum of corresponding regions' contrast according to the spatial distance among them. For this, we first find the histogram of each region r_k and then calculate the saliency s_k of r_k as

$$s_k = \sum_{(i \neq k)} n_i \cdot D_{lab}(r_k, r_i) \cdot e^{-E(r_k, r_i)/\sigma_s^2}, \tag{3.1}$$

where n_i is the number of pixels inside region r_i and $D_{lab}(r_k, r_i)$ is the color distance between regions r_k and r_i in LAB color space, $E(r_k, r_i)$ is the Euclidean spatial distance between centers of r_k and r_i, and σ_s controls the strength of spatial weighting. $\sigma_s = \frac{\sum_{j=1}^{N} dp_j}{N}$ is the average of differences between pixels pairs of the frame F_t, where dp_j is the average of absolute intensity differences between pixel p_j and its four neighbors and N is the number of pixels in F_t. We calculate the number of regions K using the super-pixel segmentation, which groups pixels of B_t into regions with similar values. With $\{s_k\}$, each pixel p_l of B_t has a saliency value. To reduce complexity, we apply a saliency thresholding of s_k to get the binary mask b_l

$$b_l = \begin{cases} 1 & : s_k(p_l) > t_s, \\ 0 & : otherwise \end{cases} \tag{3.2}$$

with t_s global to B_t defined as

$$t_s = \frac{\sum_k (n_k \cdot s_k)}{\sum_k n_k}, \tag{3.3}$$

where n_k is the number of pixels in r_k. Finally, our drift detector determines whether the target object inside B_t has drifted from its expected position depending on α_s, the ratio of the binary salient pixels inside $\{b_l\}$, as follows

$$TDrift = \begin{cases} 1 & : (\alpha_s < c_{s1}) \vee ((\alpha_s > c_{s2}) \wedge (\alpha_s < c_{s3})) \\ 0 & : otherwise. \end{cases} \tag{3.4}$$

Meaning if α_s the ratio of binary pixels in B_t is within the range (c_{s2}, c_{s3}), such as 0.6 and 0.9, or smaller than c_{s1}, such as 0.2, then drift is detected with

$$\alpha_s = \frac{N_b}{\sum_k n_k}, \tag{3.5}$$

where N_b is the number of binary pixels (i.e., pixels p_l with $b_l = 1$) of B_t, $\sum_k n_k$ is the total number of pixels in all K regions (or the number of pixels in B_t). c_{s1}, c_{s2}, and c_{s3} are experimentally selected constants that decide whether the target saliency is low and hence a drift starts to occur. As shown in Figure 3.2, such constants are selected according to tracker's scale property. Scale variant trackers, such as DSST [13], SAMF [11] , and STAPLE [3], adapt the estimated BB to the target size (Figure 3.2. a), while scale-invariant ones, such as KCF [10] and STRUCK [2], have a fixed size estimated BB (Figure 3.2. b).

(a) Scale-variant.

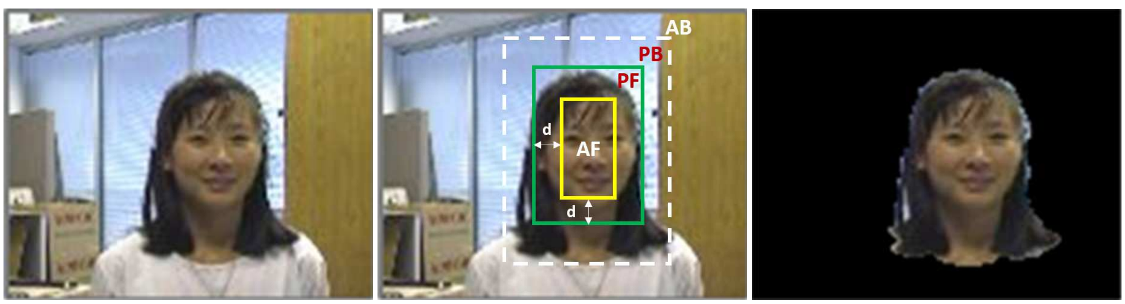
(b) Scale-Invariant.

Figure 3.2: Scale-variant versus scale-invariant example frames.

3.5.2 Drift Correction Using Seeded Segmentation

To correct drift, we relocate B_t using seeded G-Cut segmentation that has appealing results as it compromises between the computational complexity and the ability to achieve global solution [78]. The input parameters of interactive seeded segmentation [75–78], such as G-Cut, are seeds that represent both object and background as hard constraints through user interaction. Interactive seed selection has shown to improve the tracking quality [29]. However, interactivity is impractical for automated tracking applications. We thus propose to automatically select seeds for G-Cut using two-layer filter: SIFT interest points and non-background pixels inside B_t.

SIFT is able to find distinctive interest points that are invariant to location, scale and rotation, and robust to affine transformation and illumination changes [90]. Among the SIFT points, there exist interest points that are more likely to belong to the background (not the target) which may mislead the segmentation. Accordingly, we propose a two-layered filter such that SIFT interest points used to initialize the segmentation are more likely to belong to the target. The first layer uses the already generated saliency map $\{s_k\}$ to filter out all points outside the saliency map of the target object. We select only those interest points that intersect with the most salient pixels of the binarized saliency map b_l. The second layer filters out the interest points that belong to the background. We define the background as illustrated in Figure 3.3; Given the base tracker BB (green), we divide the frame into four regions: absolutely foreground (AF), probably foreground (PF), probably background (PB), and absolutely background (AB). The AF region is inside BB and belongs to the object as a hard constraint. The PB is a margin to handle segmentation of irregular object parts outside the BB. Seeded segmentation expands from AF through PF (and possibly PB) regions until it reaches BB boundaries. In the proposed method, PB and AB outside the BB are considered background. Thus the final interest points exclude those in PB and AB.



(a) Input frame. (b) Seed masks. (c) Segmentation.

Figure 3.3: Automatic seed masks for segmentation.

Seeded segmentation is sensitive to seed quantity and placement. It is important to select seeds that have a low probability of false alarm. As a consequence, our seeds selection in region AF avoids boundaries of BB and places the seeds starting from the center of the output BB of a tracker. Low-light (or dark) objects surrounded with a dark background are a challenge for segmentation and we thus select less seeds for such BB to decrease false alarm. To this end, we use the average intensities μ_o of AF region and μ_b of its immediate neighbour pixels (e.g., in a radius of 10 pixels). The idea is to determine the appropriate AF region inside B_1 centered at a distance d from its boundaries as

$$d = \begin{cases} (1 - 0.1\mu_o) \cdot \upsilon & : (\mu_o < c_o) \wedge (|\mu_o - \mu_b| < c_\mu) \\ (1 - 2\mu_o + \mu_o^2) \cdot \upsilon & : otherwise, \end{cases} \tag{3.6}$$

where $\upsilon = min(W, H)/2$ with W and H as the width and height of B_1. Then d is upper bounded to $\upsilon - 1$, which is the maximum distance to move from any B_1 boundary to reach the center of B_1. $c_o = 0.35$ represents dark objects and $c_\mu = 0.03$. Dark objects on low intensity background are a challenge for segmentation; we thus assume that dark objects on low intensity background, e.g., Figure 3.14 (third row), require lower AF region (i.e., less seeds) for accurate segmentation. This is because lower AF at the B_1 center allows the segmentation, not the seeds, to decide what are the object parts inside B_1. Figure 3.4 shows the relation between d (in pixels) and μ_o: for high μ_o (bright BB), small d (i.e., more seeds) are required as the object boundaries are more distinguishable.

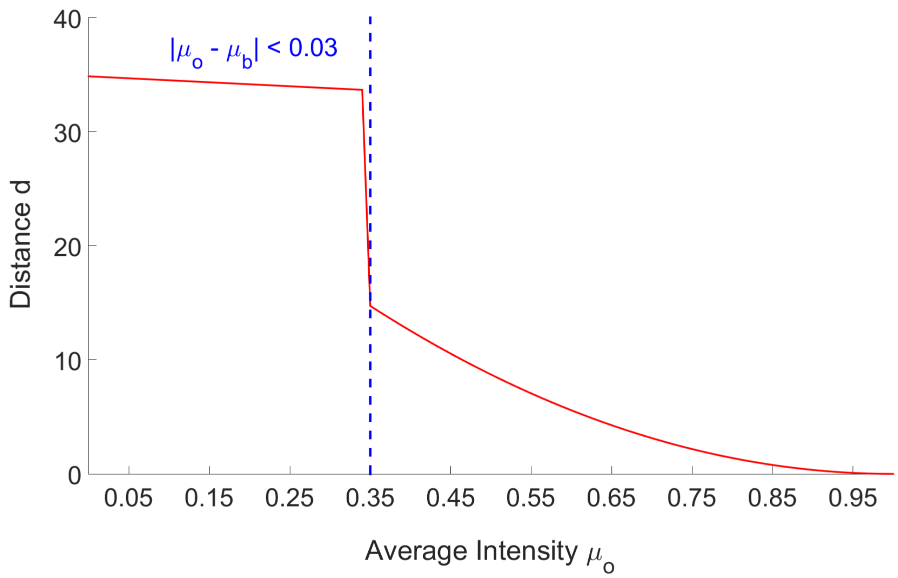


Figure 3.4: Distance d versus the average intensity μ_o of AF region.

Thus the output of our two-layer filter are seeds S that is a set of pixels p_l in B_t that are SIFT interest points filtered by the binary saliency map and at the same time fall inside the AF region,

$$S = \{p_l \in (SIFT \wedge AF) \wedge (b_l = 1)\}. \tag{3.7}$$

AF is inside B_t and defined by d in (3.6) as illustrated in Figure 3.3. Figure 3.5 illustrates the filtering process of SIFT points and segmentation result.

(a) Input BB. (b) SIFT points. (c) Filtered SIFT. (d) Segmentation.

Figure 3.5: Interest points filtering process and segmentation output.

The object segmentation initialized with seeds S produces the object mask M that represents the target object. Due to various tracking and segmentation challenges, the object mask M from G-Cut is likely to include noisy blobs. Thus we apply contour selection, that uses the flood filling to fill small holes in M through connected component algorithm, calculates the contour length of all regions inside M, selects the largest contour as C, and removes other small blobs. The BB is then relocated around the center of C.

3.6 Results and Analysis

3.6.1 Experimental Setup

For experiments, we have evaluated the results of our approach on a publicly available dataset of 100 sequences provided by Wu et al. [56] that covers 11 different tracking challenges. We test our approach on five recent trackers, STRUCK, KCF, SAMF, DSST, and STAPLE [2, 3, 10, 11, 13], that are different performing from view point of tracking accuracy, performance, and methodology [55–57, 91–95].

For evaluation, we use the average precision and success plots, and three evaluation measures: overlap ratio AOR, center location error CLE, and failure rate FR. Moreover, we use the number of recovery and drift in the form of recovery-drift plot to further investigate the achieved improvement by the proposed method (recovery and drift measures are defined in section 6.5.2).

The suitable selection of G-Cut segmentation parameters plays an important role in the accuracy of the resulting segmentation. These parameters are λ that represents a weighting term to control both over and under segmentation and σ that represents the camera noise. We propose to select such G-Cut parameter using CLONALG-AIS as described in chapter 5. In all the simulations that follows, we used the values derived in chapter 5, which are $\lambda = 218$ and $\sigma = 10$.

The only parameters that we use depending on the tracker category are those in (3.4) which we selected according to the scale property of the tracker as follows: $c_{s1} = 0.375$, $c_{s2} = 0.625$, and $c_{s3} = 0.925$ for scale-variant trackers (such as SAMF, DSST, and STAPLE) and $c_{s1} = 0.2$, $c_{s2} = 0.6$, and $c_{s3} = 0.9$ for scale-invariant trackers (such as STRUCK and KCF).

3.6.2 Objective Results

Tables 3.1, 3.2, 3.3, 3.4, and 3.5 show the overlap ratio, center location error, failure rate, drift, and recovery measures of the original trackers vs. the proposed framework for each of the 100 test sequences. Better results are shown in bold. Averages over all test sequences are also given. As can be seen, the proposed method improves the quality in all aspects. In table 3.5, we give the a pure recover-to-drift measure $pRD = \frac{(R-D)}{(L-100)}$ where R is the sum of individual recoveries of all 100 test videos, D is the sum of individual drifts of all 100 test videos, and L is the total number of frames of all test videos. Note that we subtract 100 as we skip the first frame in each video. As can be seen, with our method all trackers achieved better pRD and SAMF tracker achieved the best improvement. Note that pRD is between 1 and -1, where positive values mean the tracker well recovered from drifts on average. For example, $pRD = 0.333$ indicates good performance since the tracker shows more recoveries than drifts.

Table 3.1: The overlap ratio of base and modified trackers per sequence.

Sequence	Tracker									
	STRUCK	STRUCK-SegTrack	KCF	KCF-SegTrack	SAMF	SAMF-SegTrack	DSST	DSST-SegTrack	STAPLE	STAPLE-SegTrack
Deer	0.740	0.740	0.622	**0.687**	0.671	**0.710**	0.642	**0.805**	0.641	**0.783**
Shaking	0.448	0.443	0.039	**0.660**	0.225	**0.735**	0.716	**0.724**	0.039	**0.730**
Sylvester	0.732	**0.735**	0.643	0.625	0.629	**0.649**	0.628	**0.632**	0.566	**0.570**
David	0.226	**0.301**	0.538	0.527	0.713	**0.714**	0.818	0.808	0.794	0.761
Walking2	0.510	0.508	0.395	**0.454**	0.662	**0.680**	0.800	0.800	0.777	0.769
Car4	0.490	0.487	0.483	0.482	0.747	0.729	0.896	0.839	0.873	0.833
Girl	0.741	**0.744**	0.545	0.540	0.674	0.653	0.441	**0.448**	0.503	**0.538**
Trellis	0.612	**0.623**	0.631	0.629	0.846	**0.798**	0.773	0.770	0.845	0.816
Biker	0.254	0.237	0.249	0.243	0.244	**0.299**	0.274	0.273	0.256	**0.344**
Dudek	0.654	**0.664**	0.727	**0.728**	0.823	0.808	0.788	**0.792**	0.656	**0.730**
Human9	0.113	**0.148**	0.393	0.389	0.321	**0.339**	0.319	0.319	0.454	0.369
BasketBall	0.058	0.043	0.676	0.676	0.544	0.525	0.578	**0.594**	0.694	**0.751**
Bird1	0.136	**0.168**	0.052	**0.243**	0.165	**0.209**	0.104	**0.256**	0.286	0.286
BlurBody	0.722	0.697	0.672	**0.674**	0.690	**0.721**	0.461	**0.462**	0.727	0.711
BlueCar2	0.753	0.709	0.759	0.742	0.868	0.822	0.903	0.872	0.890	0.875
BlurFace	0.593	**0.601**	0.836	**0.848**	0.869	**0.881**	0.872	**0.888**	0.849	**0.871**
BlurOwl	0.775	**0.808**	0.194	0.193	0.666	0.666	0.184	**0.188**	0.426	**0.742**
Bolt2	0.139	0.125	0.011	**0.291**	0.011	**0.258**	0.011	**0.548**	0.681	**0.689**
Box	0.620	0.577	0.302	0.275	0.325	0.320	0.344	0.337	0.355	0.351
Car1	0.108	**0.138**	0.139	**0.167**	0.468	0.450	0.630	0.627	0.659	0.644
CarDark	0.891	0.856	0.614	0.614	0.757	0.731	0.845	0.841	0.871	**0.872**
CarScale	0.412	0.392	0.419	**0.427**	0.499	**0.550**	0.743	0.742	0.780	0.752
ClifBar	0.198	0.197	0.259	**0.400**	0.380	**0.616**	0.653	**0.656**	0.462	**0.682**
Couple	0.507	0.503	0.200	**0.348**	0.481	**0.491**	0.090	**0.449**	0.530	**0.545**
Crowds	0.088	**0.371**	0.793	**0.800**	0.735	0.691	0.732	0.732	0.803	0.763
Diving	0.292	**0.332**	0.318	**0.332**	0.236	**0.259**	0.214	**0.331**	0.244	**0.332**
DragonBaby	0.233	**0.249**	0.312	0.312	0.168	**0.180**	0.056	**0.146**	0.501	**0.538**
Football	0.324	**0.550**	0.552	**0.567**	0.599	0.589	0.553	0.553	0.582	0.566
Freeman4	0.119	**0.181**	0.175	**0.397**	0.447	0.413	0.456	**0.459**	0.408	0.384
Human3	0.014	0.014	0.005	**0.224**	0.005	**0.291**	0.023	0.023	0.024	0.023
Human4	0.208	**0.369**	0.370	**0.353**	0.671	0.626	0.645	0.643	0.666	0.584
Human6	0.200	**0.226**	0.207	0.199	0.484	0.461	0.376	**0.377**	0.805	0.779
Ironman	0.036	**0.072**	0.140	0.129	0.157	0.140	0.123	0.122	0.086	0.042
Jump	0.208	**0.260**	0.097	**0.185**	0.053	**0.159**	0.091	0.091	0.059	**0.092**
Jumping	0.640	0.569	0.274	**0.282**	0.253	**0.655**	0.136	**0.172**	0.246	**0.542**
Liquor	0.609	**0.703**	0.436	0.421	0.575	**0.730**	0.407	**0.414**	0.655	**0.764**
Matrix	0.228	**0.296**	0.119	**0.125**	0.269	0.269	0.136	0.127	0.242	**0.339**
MotorRollin	0.424	0.298	0.092	**0.112**	0.091	**0.097**	0.091	**0.098**	0.096	**0.108**
Panda	0.527	0.496	0.158	**0.264**	0.323	**0.441**	0.126	**0.127**	0.313	0.309
RedTeam	0.489	0.485	0.500	0.456	0.655	0.655	0.564	0.562	0.567	0.567
Singer2	0.043	**0.080**	0.732	0.718	0.772	0.749	0.781	0.763	0.783	0.769
Skating1	0.393	**0.412**	0.489	**0.496**	0.594	0.562	0.527	**0.536**	0.410	**0.600**
Skiing	0.034	0.031	0.050	0.050	0.050	0.055	0.065	0.065	0.105	0.102
Soccer	0.149	0.131	0.422	0.422	0.177	0.157	0.434	**0.440**	0.224	**0.555**
Surfer	0.419	0.419	0.465	**0.504**	0.677	**0.687**	0.323	**0.326**	0.219	**0.500**
Tiger2	0.367	**0.374**	0.351	0.341	0.674	**0.686**	0.325	**0.340**	0.686	**0.699**
Walking	0.566	0.559	0.530	**0.536**	0.709	0.707	0.744	0.741	0.741	0.732
Woman	0.728	0.728	0.705	**0.714**	0.687	0.638	0.693	**0.697**	0.749	**0.759**
Skating2_1	0.240	0.239	0.419	**0.434**	0.243	**0.308**	0.382	0.380	0.479	0.460
Skating2_2	0.405	0.392	0.374	0.374	0.441	0.434	0.143	**0.158**	0.254	**0.279**

Sequence	Tracker									
	STRUCK	STRUCK-SegTrack	KCF	KCF-SegTrack	SAMF	SAMF-SegTrack	DSST	DSST-SegTrack	STAPLE	STAPLE-SegTrack
Face_Occ2	**0.765**	0.740	**0.751**	0.737	0.753	**0.768**	**0.780**	0.775	0.762	**0.782**
Singer1	0.358	0.358	0.355	**0.355**	**0.539**	0.536	0.824	**0.831**	**0.822**	0.813
Boy	0.763	**0.770**	0.777	**0.799**	0.780	**0.782**	**0.839**	0.835	**0.819**	0.787
Board	**0.647**	**0.645**	0.648	**0.649**	**0.685**	0.653	**0.718**	0.710	0.603	**0.692**
BlurCar1	**0.802**	0.793	**0.802**	0.798	**0.820**	0.814	**0.767**	0.758	0.527	**0.537**
Dancer2	**0.752**	0.727	0.773	**0.781**	0.775	**0.784**	**0.771**	0.768	**0.784**	0.783
Car2	**0.687**	0.684	0.683	**0.686**	**0.855**	0.836	**0.912**	0.908	0.858	**0.919**
Dog1	0.543	0.543	**0.550**	0.540	**0.688**	0.685	0.757	**0.760**	**0.817**	0.733
FaceOcc1	0.723	**0.769**	0.774	**0.784**	**0.791**	0.786	0.765	**0.772**	**0.793**	0.527
Freeman1	0.370	**0.374**	0.214	**0.229**	**0.262**	0.259	0.244	**0.247**	**0.657**	0.646
MountainBike	0.701	**0.702**	0.711	**0.715**	0.700	**0.714**	0.730	**0.731**	0.717	**0.753**
Skater2	0.509	**0.562**	0.566	**0.574**	0.564	**0.574**	0.517	**0.529**	0.425	**0.482**
Skater	**0.621**	0.612	0.610	0.610	**0.606**	0.601	0.586	**0.588**	0.604	**0.614**
Gym	0.248	**0.254**	**0.426**	0.424	0.501	**0.477**	0.253	0.253	**0.337**	0.257
Bird2	0.559	**0.566**	0.575	**0.671**	**0.605**	0.581	0.458	**0.459**	0.781	**0.783**
BlueCar3	**0.821**	0.811	0.810	**0.817**	0.856	**0.864**	0.841	**0.844**	0.831	**0.839**
BluCar4	0.841	0.841	**0.839**	0.758	**0.859**	0.853	**0.897**	0.895	0.888	**0.894**
Bolt2	**0.139**	0.125	0.011	**0.291**	0.011	**0.258**	0.011	**0.548**	0.681	**0.689**
Car24	**0.300**	0.292	0.426	0.426	0.524	0.524	**0.467**	0.466	0.432	**0.474**
Coke	**0.556**	0.554	0.549	**0.577**	0.648	**0.655**	**0.573**	**0.585**	0.569	**0.588**
Coupon	**0.868**	0.863	0.944	0.944	0.942	0.942	**0.898**	0.897	0.899	**0.942**
Crossing	**0.646**	0.606	**0.710**	0.704	0.761	**0.770**	**0.787**	0.783	0.776	**0.778**
Dancer	**0.637**	0.643	0.646	**0.648**	0.730	**0.736**	0.771	0.771	0.776	**0.787**
David2	**0.873**	0.869	**0.827**	0.820	0.805	**0.830**	0.810	**0.812**	0.788	**0.793**
David3	**0.296**	0.293	0.772	**0.777**	**0.781**	0.779	**0.484**	0.483	0.778	**0.796**
Dog	0.328	**0.330**	**0.350**	0.342	**0.486**	0.429	**0.540**	0.543	0.528	**0.546**
Doll	0.548	**0.549**	0.534	**0.563**	**0.579**	0.571	**0.847**	0.842	0.834	**0.839**
Fish	**0.850**	0.844	**0.839**	0.756	0.825	0.825	**0.803**	0.743	**0.783**	0.780
FleetFace	**0.588**	0.558	0.589	**0.591**	0.636	**0.645**	0.634	**0.645**	**0.658**	0.635
Football1	0.324	**0.550**	0.552	**0.567**	**0.599**	0.589	0.553	0.553	**0.582**	0.566
Freeman3	**0.247**	0.204	0.324	0.324	**0.294**	0.240	0.336	0.336	0.337	**0.345**
Girl2	0.163	**0.229**	0.057	**0.250**	0.064	**0.559**	0.095	0.095	0.110	**0.123**
Human2	0.639	**0.649**	0.248	0.248	0.716	0.716	0.416	**0.429**	**0.732**	0.245
Human5	**0.343**	0.336	0.183	**0.187**	0.443	0.443	0.199	**0.316**	0.486	**0.628**
Human7	**0.483**	0.481	**0.283**	0.280	**0.327**	0.319	**0.360**	0.349	**0.810**	0.804
Human8	**0.124**	0.121	0.510	**0.511**	0.543	**0.546**	**0.807**	0.800	0.768	**0.805**
KiteSurf	0.381	**0.536**	**0.474**	0.310	**0.304**	0.273	0.322	**0.327**	**0.742**	0.704
Lemming	**0.491**	0.487	0.384	**0.408**	**0.756**	0.751	0.327	0.327	0.238	**0.360**
Man	**0.883**	0.827	0.831	**0.835**	0.818	**0.834**	**0.842**	0.831	0.852	**0.854**
Mhyang	**0.812**	0.810	**0.796**	0.793	0.865	**0.843**	0.806	**0.808**	0.779	**0.799**
Rubik	0.453	**0.454**	**0.613**	0.336	0.572	**0.574**	0.648	**0.649**	0.635	**0.681**
Subway	0.640	**0.660**	0.754	**0.769**	0.748	**0.777**	0.182	**0.183**	0.743	**0.803**
Suv	0.560	**0.577**	**0.880**	0.879	**0.858**	0.835	0.811	**0.824**	0.844	**0.845**
Tiger1	0.613	**0.616**	**0.785**	0.727	**0.782**	0.777	0.620	**0.629**	**0.764**	0.759
Toy	0.412	0.412	**0.475**	0.473	**0.612**	0.607	0.702	0.702	0.649	**0.687**
Trans	0.516	0.516	**0.384**	0.362	0.563	0.563	0.461	**0.481**	**0.551**	0.544
Twinnings	**0.586**	0.585	0.565	0.565	0.702	**0.703**	0.788	**0.797**	0.765	**0.781**
Vase	**0.309**	0.307	0.316	0.316	**0.448**	0.442	0.540	0.540	**0.580**	0.575
Jogging1	**0.672**	0.632	**0.185**	0.183	0.786	0.786	**0.185**	0.184	0.174	**0.178**
Jogging2	**0.136**	0.133	0.124	0.124	0.140	**0.269**	0.139	0.139	0.138	0.138
Average	0.465	**0.477**	0.481	**0.5**	0.559	**0.579**	0.528	**0.543**	0.586	**0.613**

Table 3.2: The center location error of base and modified trackers per sequence.

Sequence	Tracker									
	STRUCK	STRUCK-SegTrack	KCF	KCF-SegTrack	SAMF	SAMF-SegTrack	DSST	DSST-SegTrack	STAPLE	STAPLE-SegTrack
Deer	5.265	**4.978**	21.39	**9.222**	14.45	**9.864**	16.66	**4.394**	19.72	**4.905**
Shaking	27.80	47.75	112.5	**10.50**	41.86	**8.656**	8.363	**7.126**	138.0	**7.312**
Sylvester	5.879	5.890	12.91	13.27	15.16	17.60	13.52	**13.42**	14.15	14.62
David	64.92	**42.28**	8.061	9.038	3.857	4.026	3.645	3.948	3.835	**3.759**
Walking2	11.98	12.18	28.98	**23.18**	4.060	**3.202**	2.949	**2.702**	3.428	4.144
Car4	8.056	**6.635**	9.877	**9.575**	2.206	4.082	1.718	3.891	2.369	4.002
Girl	2.684	**2.671**	11.48	**11.25**	6.331	6.942	11.11	**10.35**	12.18	**11.02**
Trellis	6.125	6.328	7.763	**6.998**	2.480	5.501	2.593	2.802	3.228	**3.097**
Biker	25.55	25.77	77.17	82.00	89.56	**80.68**	74.72	74.77	79.43	**22.45**
Dudek	29.19	**28.34**	12.03	**11.42**	8.649	10.26	13.45	**12.36**	13.85	**10.67**
Human9	41.11	44.31	14.76	**13.73**	16.88	**14.30**	28.15	**26.28**	11.31	16.20
BasketBall	184.6	214.8	7.889	7.889	18.35	**18.02**	10.92	**10.41**	16.88	**5.796**
Bird1	122.4	156.2	194.8	**74.19**	97.47	108.6	144.0	154.5	65.56	**65.40**
BlurBody	15.43	18.41	14.98	14.98	18.65	**9.588**	90.85	**86.98**	6.182	7.514
BlueCar2	7.991	10.08	5.917	7.378	3.593	6.611	2.876	4.638	3.646	4.407
BlurFace	29.80	**29.42**	5.577	**4.569**	5.449	**4.813**	5.184	**4.069**	6.230	**5.033**
BlurOwl	7.043	**4.884**	183.4	**79.31**	27.35	27.35	196.1	**195.7**	123.9	**9.861**
Bolt2	376.8	377.0	6.365	6.365	4.447	5.023	4.508	**4.360**	4.047	**3.227**
Box	15.88	24.64	89.12	121.3	91.23	93.92	106.8	108.5	90.84	91.35
Car1	58.21	**33.03**	39.71	**2.175**	1.504	2.012	1.653	1.695	1.123	1.151
CarDark	1.003	1.449	6.046	6.046	2.796	3.367	1.466	1.540	1.176	**1.131**
CarScale	36.48	**36.20**	16.14	**14.11**	78.30	**10.19**	19.08	19.17	8.127	17.59
ClifBar	76.12	76.16	36.72	**14.87**	25.11	**4.464**	5.326	**5.083**	29.16	**4.588**
Couple	22.92	23.21	47.55	**22.33**	17.69	**15.89**	125.5	**30.91**	34.15	**13.90**
Crowds	374.7	**193.1**	3.069	**2.992**	3.705	4.321	3.752	**3.751**	2.890	3.067
Diving	34.76	**22.16**	39.52	**26.37**	71.30	**32.65**	73.61	**15.91**	76.48	**22.46**
DragonBaby	64.18	**60.74**	50.39	50.39	72.08	**65.08**	142.3	161.7	18.98	25.21
Football	139.4	**14.61**	14.60	**13.93**	13.36	13.58	15.75	15.75	13.00	14.14
Freeman4	51.58	**38.96**	27.11	**8.006**	10.19	**6.275**	5.603	**5.534**	18.02	17.31
Human3	231.3	240.9	260.1	**98.13**	269.7	**78.40**	332.5	**245.7**	303.7	**289.5**
Human4	234.5	**94.14**	131.7	**138.2**	4.546	8.853	5.658	5.696	5.105	12.00
Human6	102.5	**38.40**	107.6	140.8	11.93	12.57	167.3	**151.6**	5.744	7.412
Ironman	171.8	**151.5**	158.8	192.2	64.05	64.35	206.1	**205.4**	81.95	86.59
Jump	51.79	**30.71**	84.11	**52.65**	156.4	**51.84**	92.35	**89.61**	158.0	**89.26**
Jumping	5.971	7.679	26.11	**25.36**	26.95	**6.874**	36.86	**27.18**	26.68	**8.029**
Liquor	52.54	**21.45**	88.17	88.73	34.12	**16.49**	98.70	**96.22**	8.392	**7.331**
Matrix	106.8	**63.15**	76.42	**69.00**	65.95	65.95	70.05	70.29	74.14	**37.72**
MotorRolling	28.87	61.65	202.9	**175.3**	226.3	**180.1**	296.9	**142.5**	182.2	**146.7**
Panda	6.783	7.534	42.05	48.00	56.16	**9.688**	43.57	**43.56**	51.55	60.16
RedTeam	4.037	4.221	3.807	4.668	3.084	3.084	2.855	2.873	3.040	**2.705**
Singer2	171.8	**81.25**	10.28	10.82	8.104	10.67	7.772	8.802	7.597	8.066
Skating1	55.90	**36.41**	7.668	**6.548**	5.685	**5.566**	8.325	**7.889**	70.61	**6.155**
Skiing	256.8	252.8	260.0	**257.0**	251.2	253.8	195.6	195.6	244.4	**242.2**
Soccer	85.31	95.95	15.37	15.73	82.82	**55.15**	20.28	**19.62**	65.64	**9.445**
Surfer	9.495	9.495	8.737	**5.496**	4.219	**4.036**	20.06	**19.91**	27.51	**4.879**
Tiger2	46.18	**45.55**	48.42	49.14	12.13	**11.62**	41.44	**40.41**	10.67	**9.860**
Walking	3.689	**3.413**	3.970	**3.596**	2.013	2.269	1.602	1.747	1.971	1.976
Woman	3.938	3.938	10.06	11.25	9.883	11.56	9.962	10.03	2.464	2.873
Skating2_1	58.25	61.22	24.00	**23.82**	56.59	**52.34**	34.66	35.51	23.07	29.52
Skating2_2	38.40	49.02	42.10	43.13	28.98	33.78	196.1	**191.1**	68.43	**61.41**

Sequence	Tracker									
	STRUCK	STRUCK-SegTrack	KCF	KCF-SegTrack	SAMF	SAMF-SegTrack	DSST	DSST-SegTrack	STAPLE	STAPLE-SegTrack
Face_Occ2	7.079	**1.292**	7.666	**0.286**	8.429	**0.239**	6.796	**0.203**	7.942	**0.206**
Singer1	**12.17**	16.34	**10.76**	11.50	**3.160**	5.010	**3.303**	3.845	**3.384**	4.125
Boy	3.810	**3.608**	2.867	**2.292**	3.022	**2.654**	**1.969**	2.090	**2.493**	3.117
Board	**34.02**	34.25	**38.33**	38.35	**41.76**	45.64	**30.83**	32.45	**31.88**	33.94
BlurCar1	**4.885**	5.454	**4.336**	4.491	**3.746**	4.311	**4.215**	5.031	**56.33**	97.14
Dancer2	**9.395**	11.44	6.412	**6.114**	**6.244**	6.319	**6.667**	7.293	**7.302**	7.374
Car2	**3.131**	3.878	3.967	**3.805**	**1.816**	2.090	**1.523**	1.602	1.610	**1.296**
Dog1	5.692	**5.519**	**4.233**	5.076	**3.944**	4.201	4.331	**4.248**	4.595	**4.538**
FaceOcc1	18.84	**13.16**	13.90	**12.16**	11.38	**10.67**	13.77	**12.24**	**13.48**	69.27
Freeman1	11.11	**10.54**	94.88	**24.53**	61.75	**61.42**	117.7	**115.7**	**6.638**	6.985
MountainBike	9.013	**8.966**	7.661	**7.533**	8.690	**8.409**	7.763	**7.742**	8.943	**7.491**
Skater2	23.51	**17.58**	17.90	**16.59**	**19.14**	19.84	26.86	**25.54**	**15.19**	16.91
Skater	**9.081**	10.24	10.69	10.69	**9.427**	9.535	8.363	**8.115**	11.48	**9.112**
Gym	61.47	**61.12**	16.25	**15.98**	**8.118**	9.493	**13.60**	13.67	**8.593**	10.81
Bird2	20.49	**19.82**	21.37	**12.82**	**19.22**	21.05	55.57	**55.50**	6.805	**6.329**
BlueCar3	**3.229**	3.892	4.137	**3.748**	3.490	**3.187**	3.267	**2.931**	3.757	**2.887**
BluCar4	5.521	5.521	**6.487**	15.26	**5.592**	6.310	**3.846**	4.032	4.458	**3.853**
Bolt2	**110.8**	163.6	**274.2**	60.41	286.8	**61.15**	115.5	**9.267**	6.984	**6.400**
Car24	**57.33**	57.74	4.097	**2.297**	9.683	9.683	**1.696**	1.759	1.852	**1.593**
Coke	**18.21**	18.44	**18.65**	17.63	**9.791**	10.12	12.78	**12.48**	**12.19**	16.28
Coupon	**3.992**	4.330	1.568	1.568	1.597	1.597	**3.227**	3.228	2.839	**1.742**
Crossing	**3.390**	3.787	**2.249**	2.464	**1.861**	2.025	**1.430**	1.951	**1.485**	1.901
Dancer	7.922	**7.813**	**6.234**	6.244	7.493	**7.136**	**7.247**	7.417	6.400	**6.342**
David2	**1.498**	1.507	**2.082**	2.222	2.647	**2.130**	2.044	**2.021**	2.531	**2.435**
David3	**104.6**	105.8	**4.302**	4.429	**4.726**	5.094	88.25	**86.16**	3.932	**3.583**
Dog	10.80	**10.55**	**5.216**	7.652	**7.249**	7.614	7.417	**7.304**	6.882	**5.997**
Doll	8.796	**8.609**	8.322	**5.388**	4.373	**4.312**	**2.859**	3.233	**3.503**	3.612
Fish	**3.853**	4.041	**4.077**	7.475	5.338	5.338	**4.107**	6.895	**4.048**	6.160
FleetFace	**27.25**	32.17	**25.55**	25.71	24.61	**23.79**	27.58	**26.82**	**27.45**	29.47
Football1	20.88	**6.175**	5.473	**4.158**	21.91	**11.23**	9.337	**8.865**	3.161	**2.443**
Freeman3	15.51	**13.62**	19.25	19.25	**21.63**	22.61	16.39	16.39	14.56	**14.10**
Girl2	177.7	**131.5**	263.3	**135.8**	371.5	**14.39**	128.2	**127.5**	**114.1**	130.3
Human2	28.38	**28.06**	100.9	100.9	14.31	14.31	101.9	**78.25**	**12.67**	100.7
Human5	**8.306**	8.958	**174.5**	225.9	8.553	8.553	302.5	**95.70**	**4.201**	4.448
Human7	7.635	**6.532**	**47.14**	48.16	45.02	**44.33**	**25.73**	26.64	**2.231**	3.246
Human8	**74.19**	75.18	3.842	**3.619**	2.902	**2.599**	**2.232**	2.720	**2.148**	2.520
KiteSurf	28.25	**9.787**	**17.26**	51.20	38.44	**31.45**	25.36	**25.28**	**2.819**	2.973
Lemming	**76.94**	77.32	77.86	**76.50**	**7.592**	8.669	81.90	**81.46**	155.1	**79.06**
Man	**1.505**	2.412	**2.259**	2.450	2.329	**2.089**	**1.568**	1.926	**1.703**	1.850
Mhyang	**2.774**	3.068	**3.921**	4.054	**2.459**	3.608	2.213	**2.162**	2.695	**2.226**
Rubik	24.54	**24.21**	**9.383**	40.59	**15.34**	23.33	5.882	**5.836**	**4.945**	5.390
Subway	**5.380**	5.511	2.969	**2.864**	3.083	**3.079**	146.8	**140.5**	2.677	**2.099**
Suv	39.03	**28.71**	3.510	**3.042**	**3.699**	3.732	3.817	**2.970**	3.110	**2.317**
Tiger1	16.91	**16.77**	**8.052**	15.73	**6.830**	7.282	18.05	**17.61**	8.096	**7.700**
Toy	**18.82**	18.95	**7.796**	8.267	**8.201**	8.944	8.862	8.862	9.382	**8.167**
Trans	27.43	**26.70**	**93.09**	101.5	37.60	**35.62**	67.07	**61.52**	**26.43**	26.50
Twinnings	**6.731**	6.745	**6.767**	7.212	**3.582**	3.991	3.635	**3.373**	4.252	**3.881**
Vase	16.64	**16.50**	12.42	**12.31**	**12.72**	15.15	**12.21**	12.30	**11.80**	12.23
Jogging1	5.815	**5.812**	**87.89**	88.45	4.551	4.551	110.6	**101.6**	**90.73**	90.90
Jogging2	114.9	**109.1**	**137.7**	139.9	138.0	**33.05**	150.8	**136.6**	139.1	**135.3**
Average	48.94	**41.95**	44.32	**35.6**	35.04	**22.76**	48.31	**40.33**	31.36	**25.48**

Table 3.3: The failure rate of base and modified trackers per sequence.

Sequence	Tracker									
	STRUCK	STRUCK-SegTrack	KCF	KCF-SegTrack	SAMF	SAMF-SegTrack	DSST	DSST-SegTrack	STAPLE	STAPLE-SegTrack
Deer	0	0	0.154	**0.014**	0.112	**0.028**	0.084	**0**	0.098	**0**
Shaking	0.042	0.171	0.835	**0**	0	0	0	0	0.832	**0**
Sylvester	0	0	0.078	**0.076**	0.124	0.140	0.071	0.071	0.010	0.019
David	0.423	**0.175**	0	0	0	0	0	0	0	0
Walking2	0	0	0	0	0	0	0	0	0	0
Car4	0	0	0	0	0	0	0	0	0	0
Girl	0	0	0.092	**0.075**	0	0	0.064	0.064	0.1	**0.064**
Trellis	0.014	**0**	0	0	0	0	0	0	0	0
Biker	0.496	**0.492**	0.535	0.542	0.542	**0.535**	0.535	0.535	0.535	**0.478**
Dudek	0.049	0.049	0	0	0	0	0	0	0	0
Human9	0.526	**0.321**	0	0	0	0	0	0	0	0
BasketBall	0.861	0.897	0	0	0	0	0	0	0.085	**0**
Bird1	0.629	**0.618**	0.879	**0.504**	0.639	**0.5**	0.686	**0.612**	0.379	0.379
BlurBody	0	0	0.035	0.035	0.038	**0**	0.215	**0.212**	0	0
BlueCar2	0	0	0	0	0	0	0	0	0	0
BlurFace	0	0	0	0	0	0	0	0	0	0
BlurOwl	0	0	0.763	**0.424**	0.161	0.161	0.755	0.755	0.404	**0.017**
Bolt2	0.97	0.97	0	0	0	0	0	0	0	0
Box	0.015	0.045	0.431	0.543	0.484	0.498	0.529	0.530	0.505	**0.503**
Car1	0.537	**0.181**	0.238	**0**	0	0	0	0	0	0
CarDark	0	0	0	0	0	0	0	0	0	0
CarScale	0	0	0	0	0.321	**0**	0	0	0	0
ClifBar	0.544	**0.543**	0.463	**0.004**	0.046	**0**	0	0	0.262	**0**
Couple	0.167	0.167	0.664	**0.357**	0.192	0.192	0.878	**0.35**	0.307	**0.164**
Crowds	0.874	**0.442**	0	0	0	0	0	0	0	0
Diving	0.076	**0**	0.172	**0.051**	0.502	**0.162**	0.502	0	0.502	**0.009**
DragonBaby	0.314	**0.283**	0.300	0.300	0.469	**0.292**	0.867	**0.752**	0.061	0.141
Football	0.570	**0.066**	0.080	**0.066**	0.071	**0.063**	0.091	0.091	0.063	**0.019**
Freeman4	0.727	**0.512**	0.452	**0.053**	0.116	**0**	0.042	0.042	0.289	0.300
Human3	0.958	0.958	0.987	**0.541**	0.987	**0.243**	0.965	0.965	0.961	0.961
Human4	0.691	**0.436**	0.461	0.469	0.043	0.110	0.071	0.071	0.041	0.136
Human6	0.563	**0.068**	0.599	0.688	0.051	0.053	0.510	**0.506**	0	0
Ironman	0.858	**0.725**	0.722	0.722	0.536	**0.481**	0.777	0.777	0.765	**0.740**
Jump	0.176	**0**	0.622	**0**	0.893	**0**	0.729	**0.713**	0.901	**0.680**
Jumping	0	0	0.204	**0.166**	0.204	**0.041**	0.003	0.035	0.191	**0.019**
Liquor	0.240	**0.102**	0.360	**0.350**	0.171	**0**	0.290	**0.286**	0	0
Matrix	0.575	**0.325**	0.71	**0.58**	0.51	0.51	0.76	0.76	0.59	**0.27**
MotorRolling	0.073	0.225	0.762	**0.585**	0.798	**0.695**	0.798	**0.664**	0.731	**0.676**
Panda	0	0	0.544	**0.479**	0.396	**0**	0.552	**0.545**	0.382	0.407
RedTeam	0	0	0	0	0	0	0	0	0	0
Singer2	0.846	**0.598**	0	0	0	0	0	0	0	0
Skating1	0.215	**0.126**	0	0	0	0	0	0	0.362	**0**
Skiing	0.925	0.938	0.888	0.888	0.901	**0.888**	0.864	0.864	0.802	0.814
Soccer	0.608	0.696	0.015	0.015	0.607	**0.451**	0.053	**0.048**	0.635	**0.028**
Surfer	0.013	0.013	0.071	**0**	0	0	0.151	**0.143**	0.515	**0**
Tiger2	0.172	**0.167**	0.276	**0.273**	0	0	0.117	**0.115**	0	0
Walking	0	0	0	0	0	0	0	0	0	0
Woman	0	0	0.060	0.060	0.060	0.060	0.060	0.060	0	0
Skating2_1	0.308	0.343	0.059	**0.033**	0.283	**0.202**	0.141	0.141	0.010	**0**
Skating2_2	0.050	**0.016**	0.124	**0.120**	0.002	**0**	0.604	**0.513**	0.215	**0.131**

Sequence	Tracker									
	STRUCK	STRUCK-SegTrack	KCF	KCF-SegTrack	SAMF	SAMF-SegTrack	DSST	DSST-SegTrack	STAPLE	STAPLE-SegTrack
Face_Occ2	0	0	0	0	0	0	0	0	0	0
Singer1	0	0	0	0	0	0	0	0	0	0
Boy	0	0	0	0	0	0	0	0	0	0
Board	0.016	**0.015**	**0.051**	0.053	0.055	**0.054**	**0.048**	0.050	**0.047**	0.050
BlurCar1	0	0	0	0	0	0	0.002	0.002	0.292	**0.283**
Dancer2	0	0	0	0	0	0	0	0	0	0
Car2	0	0	0	0	0	0	0	0	0	0
Dog1	0	0	0	0	0	0	0	0	0	0
FaceOcc1	0	0	0	0	0	0	0	0	**0**	0.321
Freeman1	0.006	**0**	0.441	**0.138**	0.518	**0.515**	0.441	0.441	0	0
MountainBike	0	0	0	0	0	0	0	0	0	0
Skater2	0	0	0	0	0	0	0	0	0	0
Skater	0	0	0	0	0	0	0	**0**	0	0
Gym	0.379	0.379	0.110	0.110	0	0	0	**0**	0	0
Bird2	0	0	0.010	**0**	0	0	0.343	0.343	0	0
BlueCar3	0	0	0	0	0	0	0	0	0	0
BluCar4	0	0	0	0	0	0	0	0	0	0
Bolt2	**0.535**	0.597	0.969	**0.563**	0.969	**0.525**	0.972	**0**	0	0
Car24	**0.335**	0.366	0	0	0	0	0	0	0	0
Coke	0	0	0.051	0.051	0	0	0	0	**0**	0.051
Coupon	0	0	0	0	0	0	0	0	0	0
Crossing	0.016	0.016	0	0	0	0	0	0	0	0
Dancer	0	0	0	0	0	0	0	0	0	0
David2	0	0	0	0	0	0	0	0	0	0
David3	0.623	**0.621**	0	0	0	0	0.376	0.376	0	0
Dog	0.007	0.007	0	0	0	0	0	0	0	0
Doll	0.010	**0.006**	0	0	0	0	0	0	0	0
Fish	0	0	0	0	0	0	0	0	0	0
FleetFace	0	0	0	0	0	0	0	0	0	0
Football1	0.256	0	0.013	0.013	0.270	**0.013**	0	0	0	0
Freeman3	0.111	**0.060**	0.086	0.086	0.093	**0.089**	0.086	0.086	0.082	**0.080**
Girl2	0.722	**0.59**	0.926	**0.496**	0.91	**0**	0.588	**0.584**	0.626	**0.582**
Human2	0.007	**0.005**	0.373	0.373	0	0	0.414	**0.387**	**0**	0.586
Human5	0	0	**0.680**	0.725	0.071	0.071	0.755	**0.300**	0	0
Human7	0	0	0.528	0.528	**0.548**	0.552	0.512	0.512	0	0
Human8	**0.816**	0.820	0	0	0	0	0	0	0	0
KiteSurf	0.470	**0.142**	**0.214**	0.547	0.559	**0.547**	0.583	0.583	0	0
Lemming	0.216	0.216	0.309	**0.306**	0	0	0.338	**0.337**	0.654	**0.333**
Man	0	0	0	0	0	0	0	0	0	0
Mhyang	0	0	0	0	0	0	0	0	0	0
Rubik	0	**0**	0	0	0	0	0	0	0	0
Subway	0.002	0.002	0	0	0	0	0.742	0.742	0	0
Suv	0.180	**0.102**	0	0	0	0	0	0	0	0
Tiger1	0	0	**0**	0.016	0	0	0.025	0.025	0	0
Toy	0.047	0.047	0	0	0	0	0	0	0	0
Trans	0	0	0	0	0	0	0	0	0	0
Twinnings	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
Vase	0	0	0	0	0	0	0	0	0	0
Jogging1	0.009	0.009	0.742	0.745	0.009	0.009	0.755	0.755	0.755	0.755
Jogging2	**0.775**	0.780	0.817	0.817	0.798	**0.312**	0.801	0.801	**0.791**	0.794
Average	0.206	**0.164**	0.199	**0.146**	0.15	**0.098**	0.21	**0.175**	0.148	**0.108**

Table 3.4: The number of drifts of base and modified trackers per sequence.

Sequence	Tracker									
	STRUCK	STRUCK-SegTrack	KCF	KCF-SegTrack	SAMF	SAMF-SegTrack	DSST	DSST-SegTrack	STAPLE	STAPLE-SegTrack
Deer	0	0	10	**0**	7	**1**	3	**0**	6	**0**
Shaking	0	0	303	**0**	0	0	0	0	304	**0**
Sylvester	0	0	104	**101**	165	189	95	95	13	24
David	211	**108**	0	0	0	0	0	0	0	0
Walking2	0	0	0	0	0	0	0	0	0	0
Car4	0	0	0	0	0	0	0	0	0	0
Girl	0	0	45	**31**	0	0	32	32	47	**32**
Trellis	7	**0**	0	0	0	0	0	0	0	0
Biker	69	**68**	76	77	77	**76**	76	76	76	**68**
Dudek	0	0	0	0	0	0	0	0	0	0
Human9	142	**84**	0	0	0	0	0	0	0	0
BasketBall	574	663	0	0	0	0	0	0	62	**0**
Bird1	192	253	358	**203**	261	**199**	279	**250**	150	150
BlurBody	0	0	10	10	11	**0**	65	**64**	0	0
BlueCar2	0	0	0	0	0	0	0	0	0	0
BlurFace	0	0	0	0	0	0	0	0	0	0
BlurOwl	0	0	478	**221**	88	88	473	473	254	**9**
Bolt2	338	338	0	0	0	0	0	0	0	0
Box	0	0	461	628	553	574	613	614	583	**582**
Car1	543	**183**	243	**0**	0	0	0	0	0	0
CarDark	0	0	0	0	0	0	0	0	0	0
CarScale	0	0	0	0	81	**0**	0	0	0	0
ClifBar	253	**247**	210	**1**	18	**0**	0	0	121	**0**
Couple	10	10	89	**41**	24	21	122	**46**	42	**20**
Crowds	302	302	0	0	0	0	0	0	0	0
Diving	0	0	36	**9**	108	**33**	108	0	108	**1**
DragonBaby	46	**30**	31	31	47	**25**	94	**84**	5	14
Football	204	**23**	28	**22**	23	**18**	31	31	21	**6**
Freeman4	200	**135**	122	**15**	31	**0**	12	12	81	83
Human3	1626	1626	1677	**913**	1677	**413**	1639	1639	1633	1633
Human4	467	**290**	308	313	29	74	48	48	28	91
Human6	433	**50**	469	541	39	41	404	**399**	0	0
Ironman	142	**117**	120	120	81	**75**	129	129	117	**112**
Jump	28	**0**	74	**0**	109	**0**	89	**85**	110	**80**
Jumping	0	0	36	**27**	36	**12**	0	8	32	**3**
Liquor	384	**173**	623	**598**	296	**0**	500	**494**	0	0
Matrix	63	**12**	67	**53**	48	48	74	74	57	**26**
MotorRolling	0	22	124	**93**	131	**112**	131	**106**	119	**108**
Panda	0	0	540	**476**	396	**0**	549	**542**	382	407
RedTeam	0	0	0	0	0	0	0	0	0	0
Singer2	315	**214**	0	0	0	0	0	0	0	0
Skating1	81	**63**	0	0	0	0	0	0	143	**0**
Skiing	74	75	72	72	73	**72**	70	70	65	66
Soccer	243	286	5	**0**	233	**163**	19	**17**	245	**11**
Surfer	4	4	25	**0**	0	0	51	**47**	183	**0**
Tiger2	19	**13**	88	**87**	0	0	36	**35**	0	0
Walking	0	0	0	0	0	0	0	0	0	0
Woman	0	0	36	36	36	36	36	36	0	0
Skating2_1	133	150	26	**14**	126	**93**	61	61	3	**0**
Skating2_2	11	**6**	58	**56**	0	**0**	281	**242**	95	**59**

Sequence	Tracker									
	STRUCK	STRUCK-SegTrack	KCF	KCF-SegTrack	SAMF	SAMF-SegTrack	DSST	DSST-SegTrack	STAPLE	STAPLE-SegTrack
Face_Occ2	0	0	0	0	0	0	0	0	0	0
Singer1	0	0	0	0	0	0	0	0	0	0
Boy	0	0	0	0	0	0	0	0	0	0
Board	1	1	**36**	37	39	**38**	**34**	35	**33**	35
BlurCar1	0	0	0	0	0	0	1	1	215	**209**
Dancer2	0	0	0	0	0	0	0	0	0	0
Car2	0	0	0	0	0	0	0	0	0	0
Dog1	0	0	0	0	0	0	0	0	0	0
FaceOcc1	0	0	0	0	0	0	0	0	**0**	285
Freeman1	1	**0**	144	**39**	168	**167**	144	144	0	0
MountainBike	0	0	0	0	0	0	0	0	0	0
Skater2	0	0	0	0	0	0	0	0	0	0
Skater	0	0	0	0	0	0	0	**0**	0	0
Gym	0	0	85	83	0	0	0	**0**	0	0
Bird2	0	0	1	**0**	0	0	32	32	0	0
BlueCar3	0	0	0	0	0	0	0	0	0	0
BluCar4	0	0	0	0	0	0	0	0	0	0
Bolt2	152	**151**	284	**164**	284	**153**	285	**0**	0	0
Car24	**875**	1169	0	0	0	0	0	0	0	0
Coke	0	0	15	15	0	0	0	0	**0**	15
Coupon	0	0	0	0	0	0	0	0	0	0
Crossing	1	1	0	0	0	0	0	0	0	0
Dancer	0	0	0	0	0	0	0	0	0	0
David2	0	0	0	0	0	0	0	0	0	0
David3	156	**154**	0	0	0	0	95	95	0	0
Dog	0	0	0	0	0	0	0	0	0	0
Doll	47	**9**	0	0	0	0	0	0	0	0
Fish	0	0	0	0	0	0	0	0	0	0
FleetFace	0	0	0	0	0	0	0	0	0	0
Football1	14	0	1	1	20	**1**	0	0	0	0
Freeman3	33	**24**	40	40	43	**41**	40	40	38	**37**
Girl2	**1062**	1063	1390	**742**	1364	**0**	878	**870**	934	**859**
Human2	8	**0**	417	417	0	0	467	**435**	**0**	657
Human5	0	0	**480**	515	48	672	539	**213**	0	0
Human7	0	0	132	132	**136**	136	118	118	0	0
Human8	103	103	0	0	0	0	0	0	0	0
KiteSurf	41	**11**	**17**	45	46	**44**	49	49	0	0
Lemming	287	287	408	**405**	0	0	448	**447**	871	**441**
Man	0	0	0	0	0	0	0	0	0	0
Mhyang	0	0	0	0	0	0	0	0	0	0
Rubik	0	0	0	0	0	0	0	0	0	0
Subway	0	0	0	0	0	0	130	130	0	0
Suv	160	**74**	0	0	0	0	0	0	0	0
Tiger1	0	0	**0**	4	0	0	6	6	0	0
Toy	0	0	0	0	0	0	0	0	0	0
Trans	0	0	0	0	0	0	0	0	0	0
Twinnings	1	1	1	1	1	1	1	1	1	1
Vase	0	0	0	0	0	0	0	0	0	0
Jogging1	2	2	227	228	2	2	232	232	232	232
Jogging2	**248**	249	251	251	245	**95**	246	246	**243**	244
Average	103	**88**	108	**79**	72	**37**	98	**89**	76	**66**

Table 3.5: The number of recovery of base and modified trackers per sequence.

Sequence	Tracker									
	STRUCK	STRUCK-SegTrack	KCF	KCF-SegTrack	SAMF	SAMF-SegTrack	DSST	DSST-SegTrack	STAPLE	STAPLE-SegTrack
Deer	71	71	1	1	1	1	3	**71**	1	**71**
Shaking	365	365	2	**365**	365	365	365	365	0	**365**
Sylvester	1345	1345	1	1	2	0	1	1	1	**2**
David	0	**0**	471	471	471	471	471	471	471	471
Walking2	500	500	500	500	500	500	500	500	500	500
Car4	659	659	659	659	659	659	659	659	659	659
Girl	500	500	**1**	0	500	500	0	0	**3**	0
Trellis	1	**569**	569	569	569	569	569	569	569	569
Biker	0	**0**	0	0	0	0	0	0	0	0
Dudek	1145	1145	1145	1145	1145	1145	1145	1145	1145	1145
Human9	0	**14**	305	305	305	305	305	305	305	305
BasketBall	0	0	725	725	725	725	725	725	0	**725**
Bird1	44	8	1	**3**	0	**5**	**1**	0	5	5
BlurBody	334	334	2	2	2	**334**	7	**7**	334	334
BlueCar2	585	585	585	585	585	585	585	585	585	585
BlurFace	493	493	493	493	493	493	493	493	493	493
BlurOwl	631	631	4	**47**	14	14	4	4	1	**2**
Bolt2	0	0	350	350	350	350	350	350	350	350
Box	1161	1161	**40**	3	10	5	2	2	4	**3**
Car1	0	**2**	0	**1020**	1020	1020	1020	1020	1020	1020
CarDark	393	393	393	393	393	393	393	393	393	393
CarScale	252	252	252	252	0	**252**	252	252	252	252
ClifBar	3	**6**	**9**	1	4	**472**	472	472	3	**472**
Couple	0	0	4	**9**	3	6	1	**3**	1	**3**
Crowds	0	0	347	347	347	347	347	347	347	347
Diving	215	215	1	**2**	0	**2**	0	**215**	0	**1**
DragonBaby	1	0	3	3	6	**8**	4	**1**	2	2
Football	0	**1**	1	**2**	3	**5**	2	2	**2**	1
Freeman4	10	10	**6**	0	2	**283**	0	0	1	**2**
Human3	0	0	0	**7**	0	**1**	0	0	0	0
Human4	0	**1**	0	0	0	0	0	0	0	0
Human6	1	**4**	**6**	4	2	1	0	**2**	792	792
Ironman	3	**4**	0	0	**8**	5	0	0	10	**11**
Jump	0	**122**	2	**122**	0	**122**	0	**2**	0	**3**
Jumping	313	313	28	**25**	**28**	1	1	3	**28**	3
Liquor	1	**5**	4	**13**	3	**1741**	**5**	4	1741	1741
Matrix	0	**2**	4	**5**	3	3	2	2	**2**	1
MotorRolling	164	1	**1**	3	0	**2**	0	**3**	1	**3**
Panda	1000	1000	**4**	3	0	**1000**	3	**3**	0	0
RedTeam	1918	1918	1918	1918	1918	1918	1918	1918	1918	1918
Singer2	0	**5**	366	366	366	366	366	366	366	366
Skating1	0	0	400	400	400	400	400	400	2	**400**
Skiing	1	1	0	0	0	**0**	0	0	0	0
Soccer	6	2	1	1	5	**14**	2	2	**4**	0
Surfer	1	1	2	**376**	376	376	6	**7**	11	**376**
Tiger2	1	1	13	13	365	365	7	**7**	365	365
Walking	412	412	412	412	412	412	412	412	412	412
Woman	597	597	0	0	0	0	0	0	597	597
Skating2_1	0	**5**	2	2	**8**	3	6	6	2	**473**
Skating2_2	2	**2**	1	1	1	**473**	**5**	1	**7**	3

Sequence	Tracker									
	STRUCK	STRUCK-SegTrack	KCF	KCF-SegTrack	SAMF	SAMF-SegTrack	DSST	DSST-SegTrack	STAPLE	STAPLE-SegTrack
Face_Occ2	812	812	812	812	812	812	812	812	812	812
Singer1	351	351	351	351	351	351	351	351	351	351
Boy	602	602	602	602	602	602	602	602	602	602
Board	0	**0**	**0**	0	0	**0**	**0**	0	**0**	0
BlurCar1	742	742	742	742	742	742	1	1	2	**1**
Dancer2	150	150	150	150	150	150	150	150	150	150
Car2	913	913	913	913	913	913	913	913	913	913
Dog1	1350	1350	1350	1350	1350	1350	1350	1350	1350	1350
FaceOcc1	892	892	892	892	892	892	892	892	**892**	2
Freeman1	1	**326**	0	**6**	1	**1**	0	0	326	326
MountainBike	228	228	228	228	228	228	228	228	228	228
Skater2	435	435	435	435	435	435	435	435	435	435
Skater	160	160	160	160	160	160	160	**160**	160	160
Gym	767	767	0	0	767	767	767	**767**	767	767
Bird2	99	99	0	**99**	99	99	2	2	99	99
BlueCar3	357	357	357	357	357	357	357	357	357	357
BluCar4	380	380	380	380	380	380	380	380	380	380
Bolt2	**0**	0	350	**350**	350	**350**	350	**350**	350	350
Car24	**0**	0	3059	3059	3059	3059	3059	3059	3059	3059
Coke	291	291	0	0	291	291	291	291	**291**	0
Coupon	327	327	327	327	327	327	327	327	327	327
Crossing	1	1	120	120	120	120	120	120	120	120
Dancer	225	225	225	225	225	225	225	225	225	225
David2	537	537	537	537	537	537	537	537	537	537
David3	1	**1**	252	252	252	252	0	0	252	252
Dog	127	127	127	127	127	127	127	127	127	127
Doll	1	**3872**	3872	3872	3872	3872	3872	3872	3872	3872
Fish	476	476	476	476	476	476	476	476	476	476
FleetFace	707	707	707	707	707	707	707	707	707	707
Football1	1	74	0	0	0	**0**	74	74	74	74
Freeman3	3	**4**	0	0	0	**0**	0	0	0	**0**
Girl2	0	**0**	0	**2**	1	**1500**	5	**6**	5	**14**
Human2	1	**1128**	4	4	1128	1128	0	**2**	**1128**	5
Human5	713	713	**5**	2	3	0	0	**1**	713	713
Human7	250	250	0	0	**1**	2	10	10	250	250
Human8	**0**	0	128	128	128	128	128	128	128	128
KiteSurf	0	**1**	**1**	1	1	**2**	0	0	84	84
Lemming	0	0	6	**5**	1336	1336	4	**4**	4	**5**
Man	134	134	134	134	134	134	134	134	134	134
Mhyang	1490	1490	1490	1490	1490	1490	1490	1490	1490	1490
Rubik	1997	**1997**	1997	1997	1997	1997	1997	1997	1997	1997
Subway	175	175	175	175	175	175	0	0	175	175
Suv	5	**9**	945	945	945	945	945	945	945	945
Tiger1	354	354	**354**	2	354	354	3	3	354	354
Toy	271	271	271	271	271	271	271	271	271	271
Trans	124	124	124	124	124	124	124	124	124	124
Twinnings	0	0	0	0	0	0	0	0	0	0
Vase	271	271	271	271	271	271	271	271	271	271
Jogging1	1	1	1	1	1	1	0	0	0	0
Jogging2	**0**	0	0	0	0	**1**	0	0	**0**	0
Average R	298	**357**	330	**346**	389	**451**	344	**349**	399	**405**
Average pRD	0.331	**0.456**	0.376	**0.453**	0.537	**0.702**	0.417	**0.441**	0.548	**0.575**

In the following results and to make figures clear, we systematically select the best three trackers according to Tables 3.1, 3.2, 3.3, 3.4, and 3.5 for comparison. The average success and precision plots shown in Figures 3.6 and 3.7, confirm that the proposed method outperforms the base trackers, for all sequences in both measures. SAMF (black), KCF (blue), and STAPLE (green) trackers show the best enhancement.

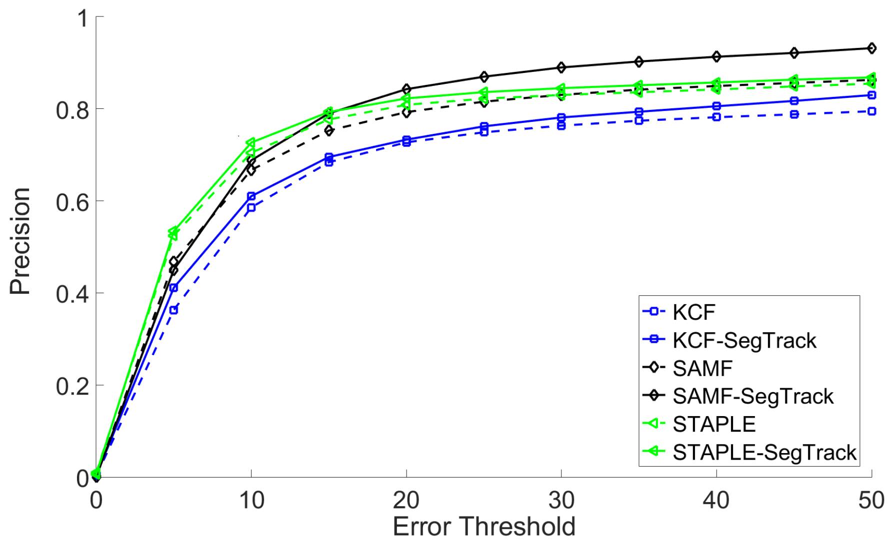


Figure 3.6: Average precision plot using automatic drift detection and correction.

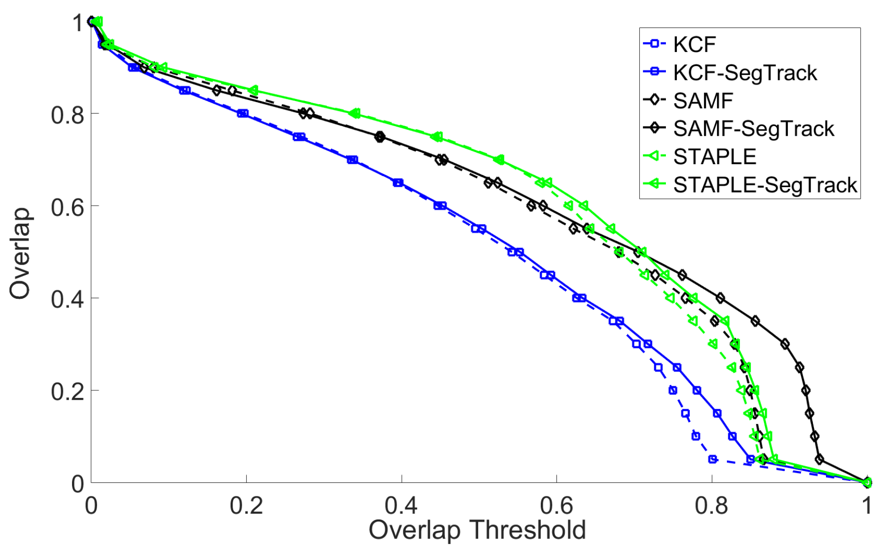
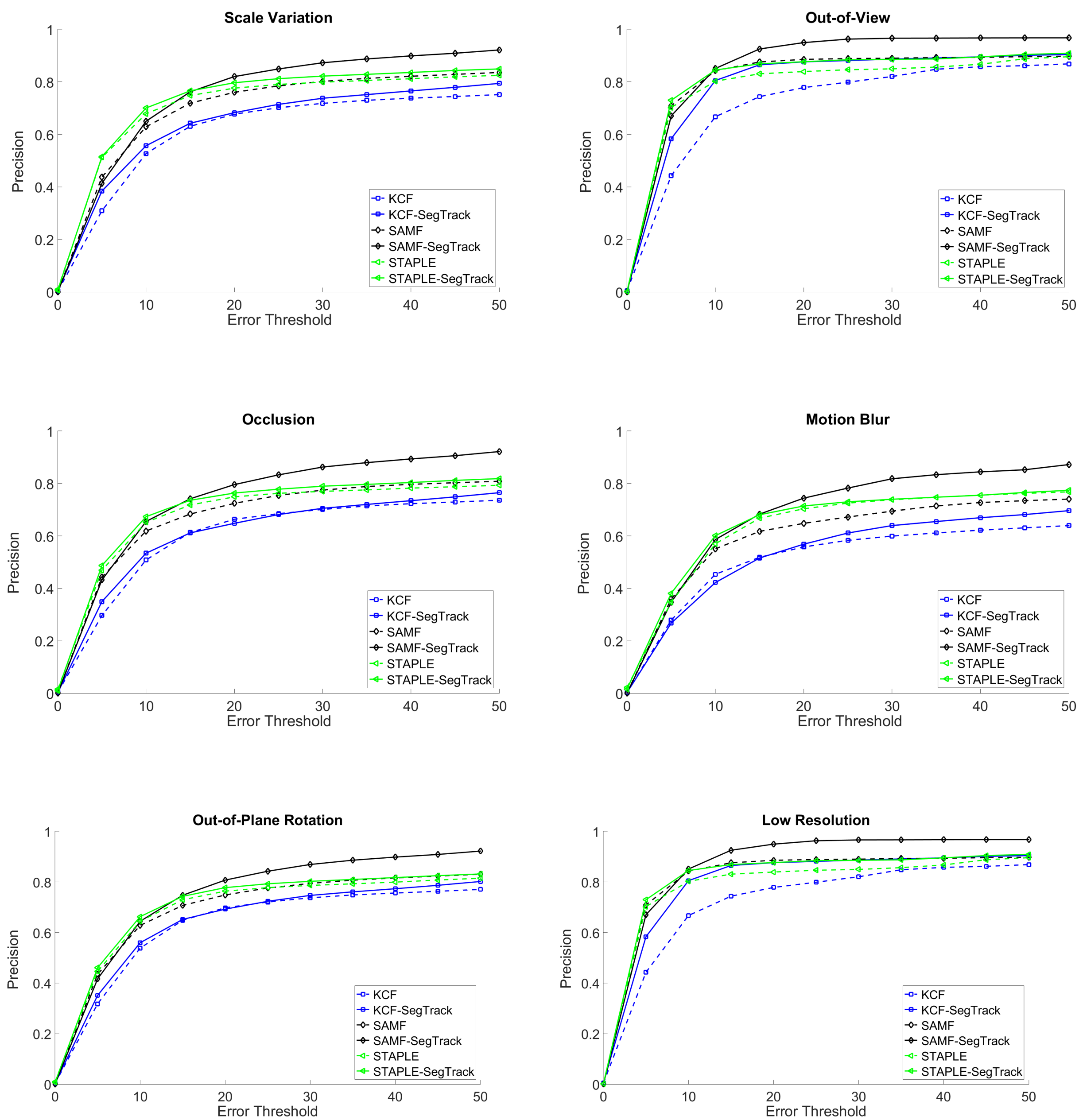


Figure 3.7: Average success plot using automatic drift detection and correction.

Figures 3.8 and 3.9, show the effectiveness of the proposed method on various challenge attributes, such as fast motion, occlusion, background clutter, etc. It is clear that the performance of the proposed method outperforms the corresponding original ones, as they effectively handle the different challenging situations.



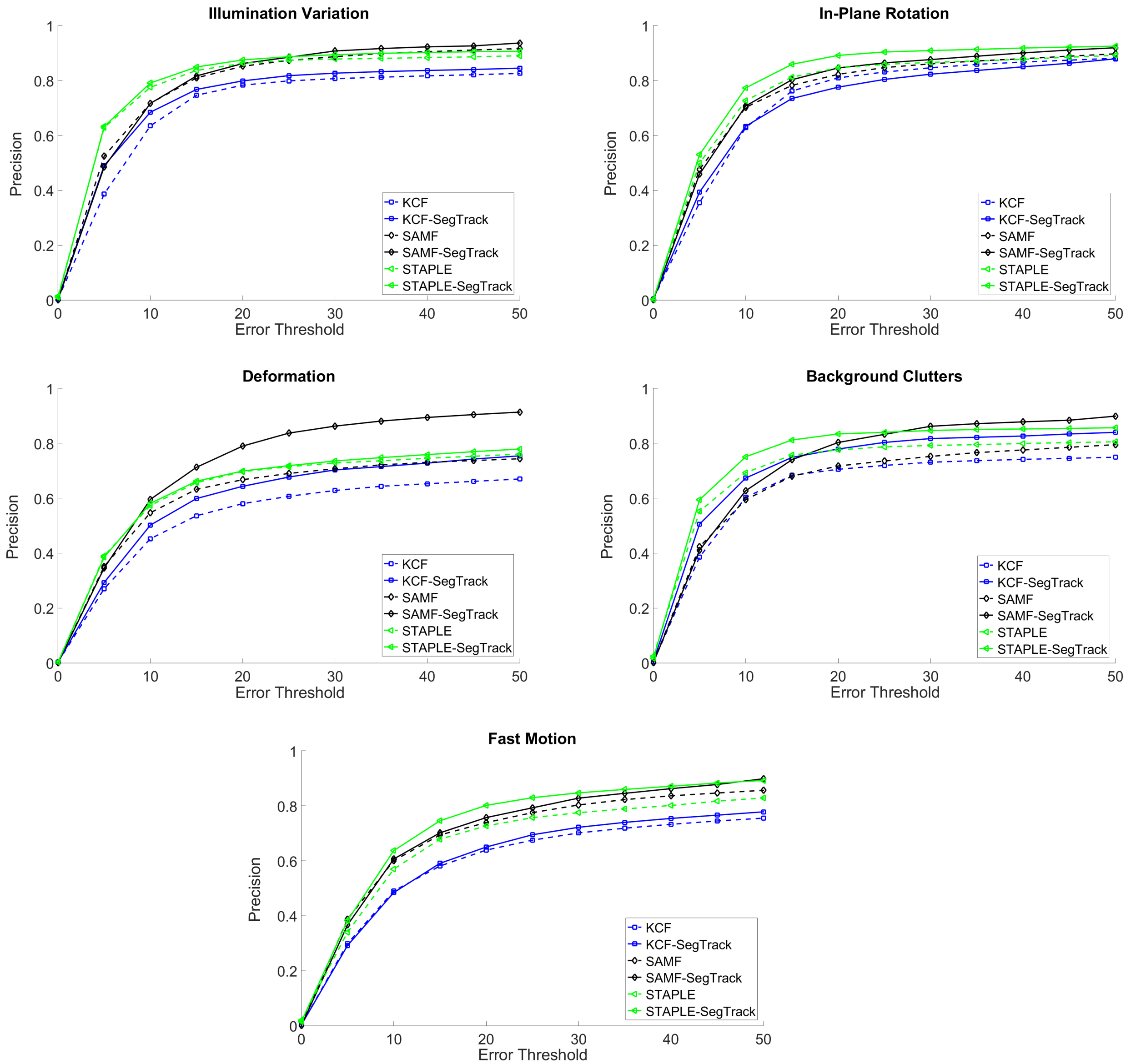
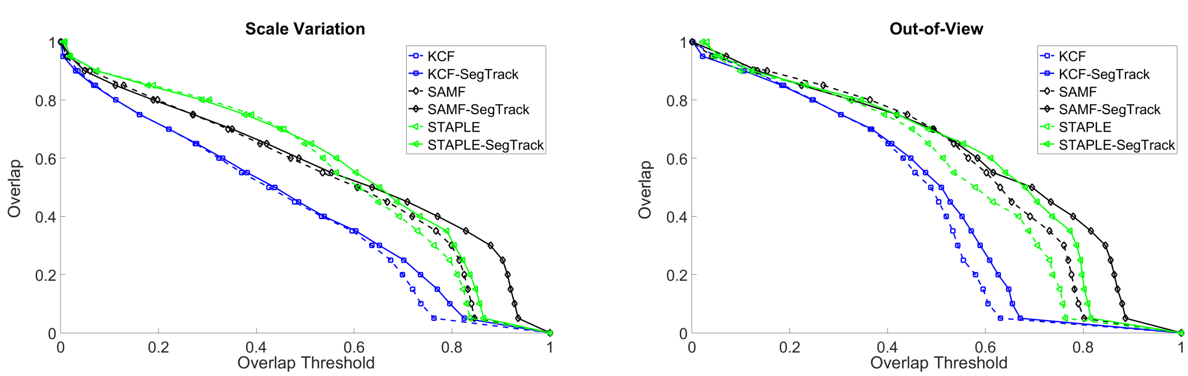
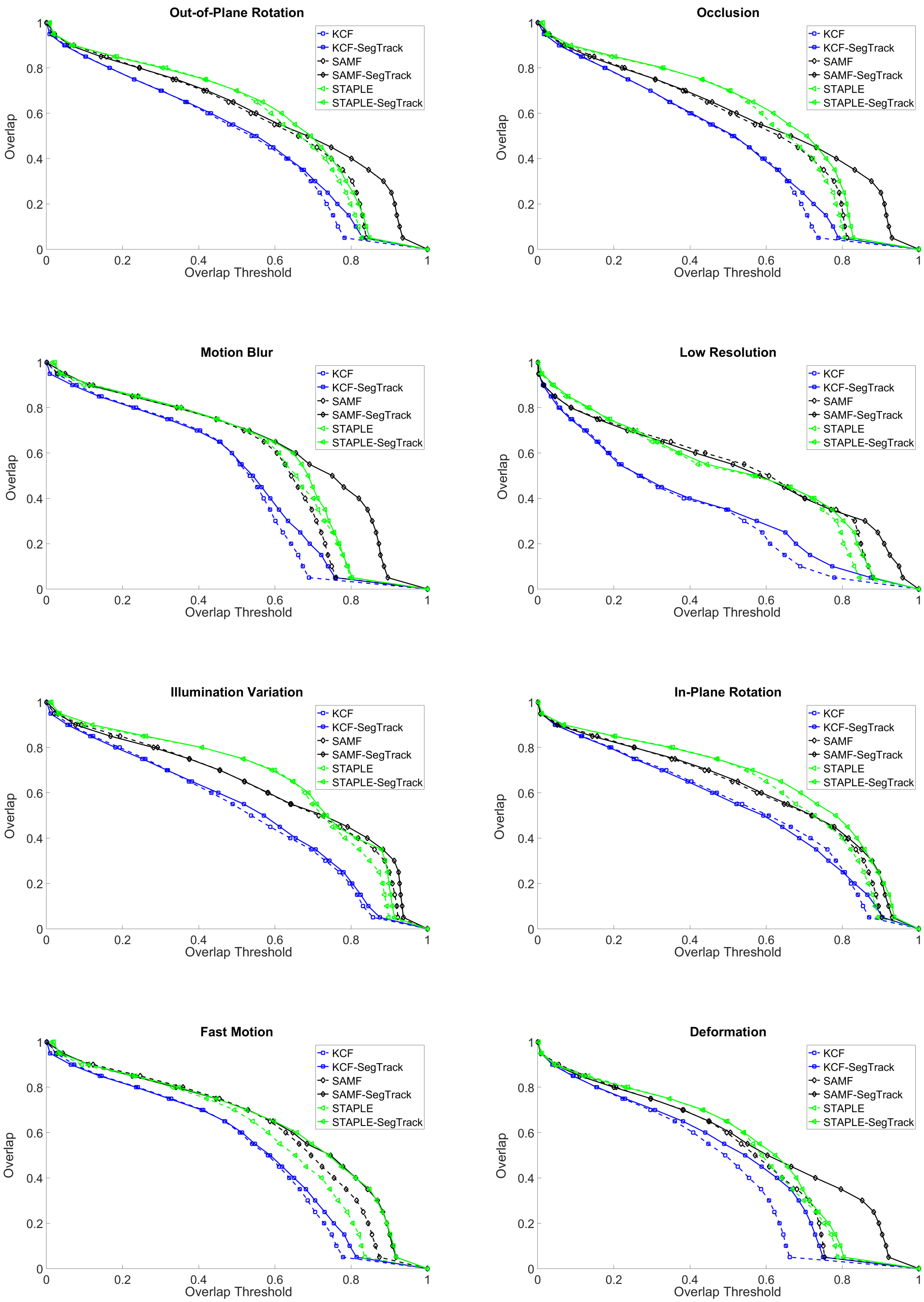


Figure 3.8: Precision plot of all trackers over all test videos per challenge.




Out-of-Plane Rotation
Occlusion
Motion Blur
Low Resolution
Illumination Variation
In-Plane Rotation
Fast Motion
Deformation
KCF
KCF-SegTrack
SAMF
SAMF-SegTrack
STAPLE
STAPLE-SegTrack
Overlap
Overlap Threshold
0
0.2
0.4
0.6
0.8
1

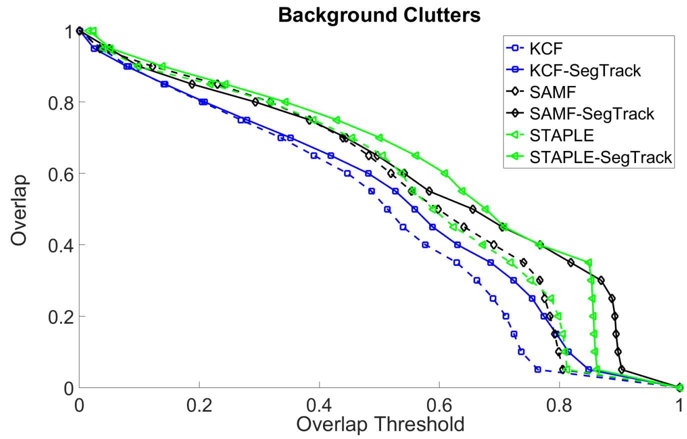


Figure 3.9: Success plot of all trackers over all test videos per challenge.

It is worth noting that, although the quality of STAPLE base tracker outperforms that of the remaining four trackers under test, according to VOT2016 [96], we found that the quality of the modified SAMF tracker outperforms that of STAPLE tracker for different challenges such as scale variation, low resolution, out-of-plane rotation, and occlusion as shown in Figures 3.8 and 3.9. In addition, the quality of the modified STRUCK tracker is better than that of DSST tracker, which is originally better than STRUCK, for low resolution, fast motion, and occlusion challenges.

Figures 3.10, 3.11 , and 3.12 show the improvement in AOR, CLE, and FR, by the proposed method applied to the all trackers over all 100 test sequences. As can be seen, the quality of our proposed method (continuous lines) outperforms that of the corresponding original trackers (dashed lines) for AOR, CLE, and FR.

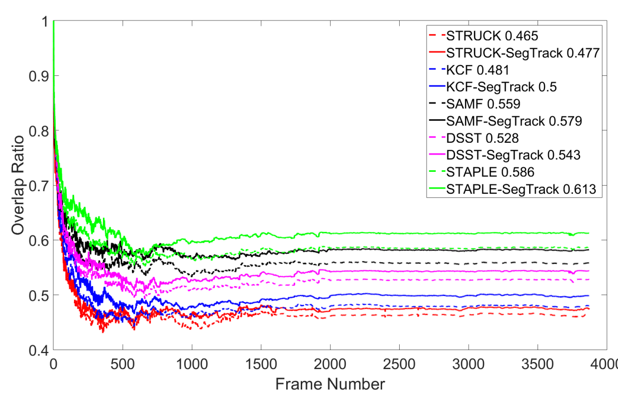


Figure 3.10: AOR of base and modified trackers over all sequences.

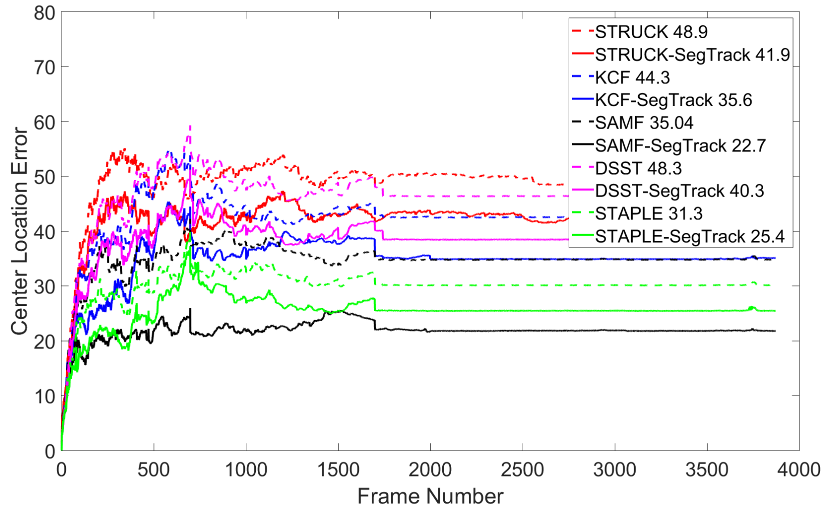


Figure 3.11: CLE of base and modified trackers over all sequences.

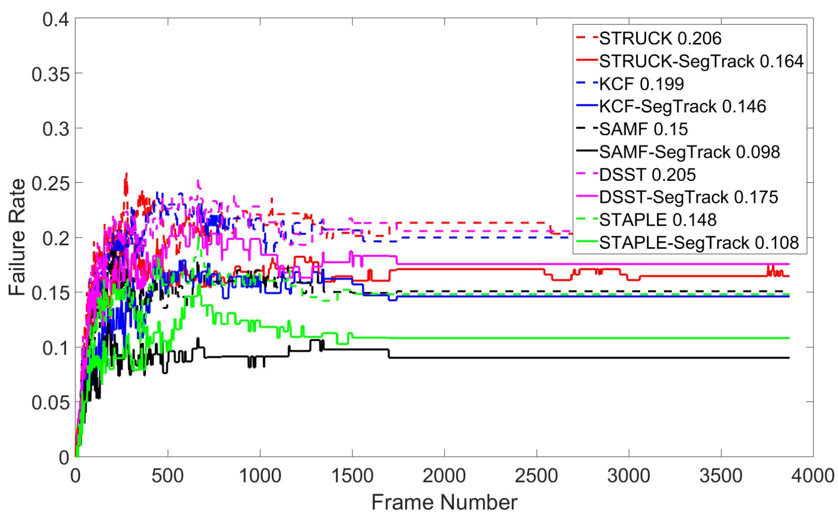


Figure 3.12: FR of base and modified trackers over all sequences.

For more investigation, we calculated the number of times that each tracker recovers from drift (or failure). We defined drift and recovery measures in section 6.5.2. In case of successful tracking without failures (B_t has overlap with the ground truth BB B_t^g) for certain sequence, we consider the tracker has recoveries by the total number of frames of that sequence). In addition, the pure number of failures for each tracker is calculated as the difference between number of recoveries and number of failures. As shown in Figure 3.13, the proposed method (filled symbols) has achieved higher recovery numbers, compared with the corresponding original trackers (clear symbols), that in turn leads to lower tracking failures. In addition, this plot facilitates the evaluation of the achieved improvement of the proposed method in both tracking failures and number of recoveries. It shows that while the quality of the original SAMF and STAPLE trackers is better than that of KCF, the proposed method decreased the failures of the KCF tracker (KCF-SegTrack) that becomes lower than that of both SAMF and STAPLE. Also STRUCK-SegTrack has achieved lower number of failures than KCF and DSST trackers which are originally better than STRUCK.

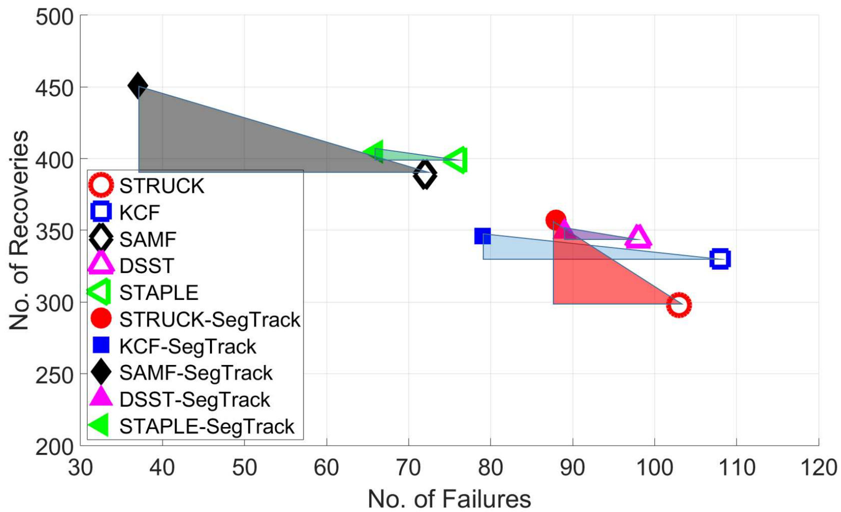


Figure 3.13: Recovery-failure (drift) plot of base and modified trackers.

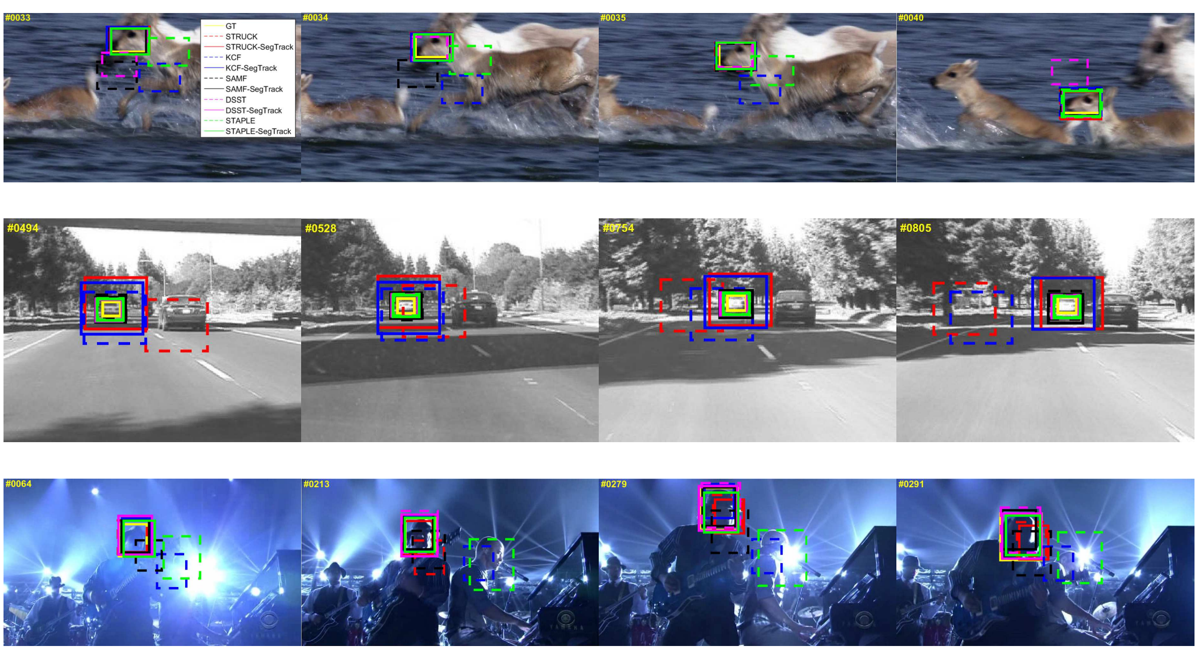
Table 3.6 summarizes the average improvement in AOR, CLE, and FR, using the proposed method (better quality is shown in bold), for the five trackers relative to their corresponding original tracking quality over all test sequences. Over all frames, the proposed method gives better quality; no outliers were noted. As can be seen, all trackers have achieved better tracking quality compared with their corresponding original versions. Hightest improvement is achieved with SAMF followed by STAPLE tracker. We notice the ranking of the five trackers based on AOR remains the same, while based on FR, the ranking is changed (best SAMF and worse DSST).

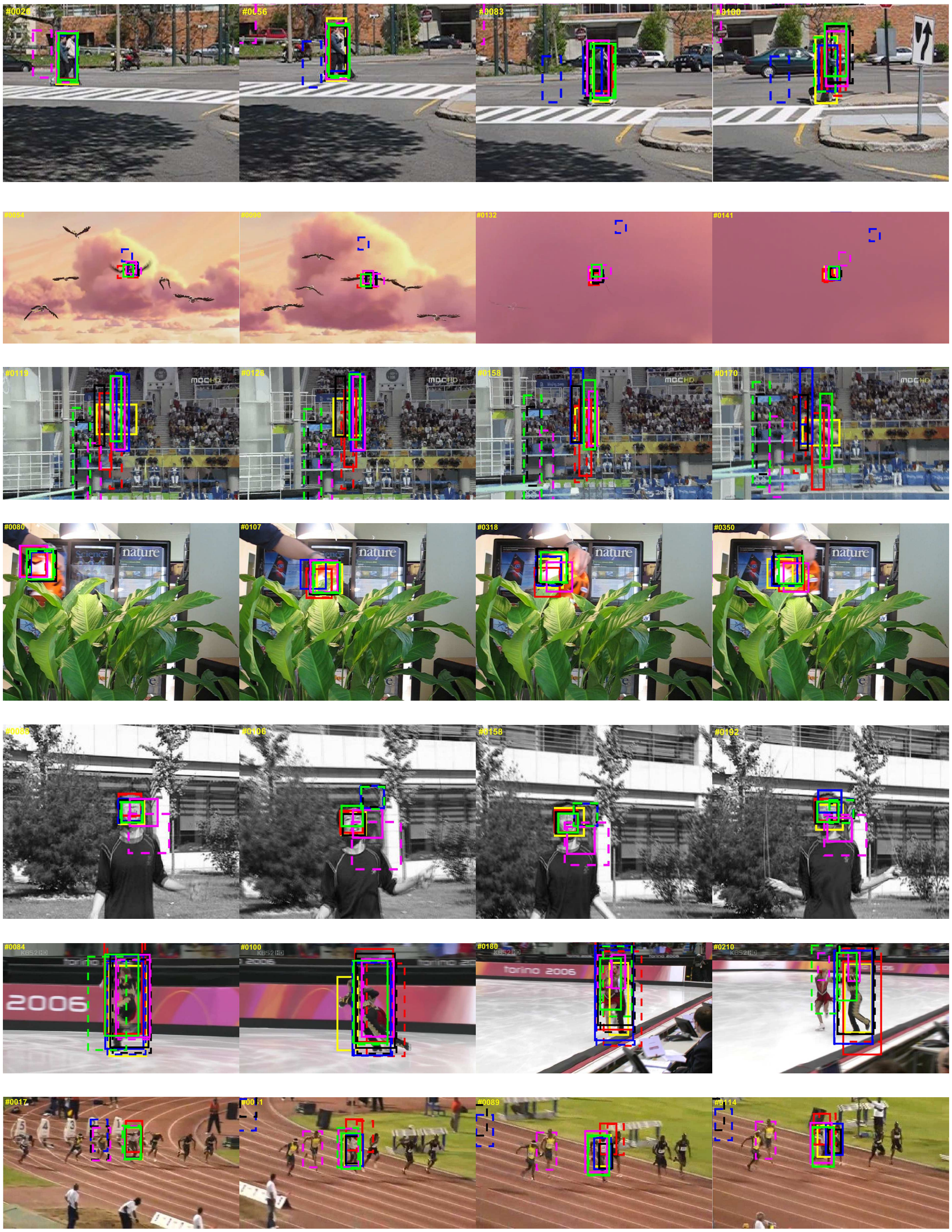
Table 3.6: Average improvement of the proposed drift detection and correction method.

Trackers	Overlap Ratio (AOR)	Center Location Error (CLE)	Failure Rate (FR)
STAPLE	0.586	31.3	0.148
STAPLE-SegTrack	**0.613**	**25.4**	**0.108**
Improvement	4.51%	18.72%	26.98%
SAMF	0.559	35.04	0.15
SAMF-SegTrack	**0.579**	**22.7**	**0.098**
Improvement	3.49%	35.03%	34.47%
DSST	0.528	48.3	0.205
DSST-SegTrack	**0.543**	**40.3**	**0.175**
Improvement	2.97%	16.52%	14.61%
KCF	0.481	44.3	0.199
KCF-SegTrack	**0.500**	**35.6**	**0.146**
Improvement	3.96%	19.66%	26.92%
STRUCK	0.465	48.9	0.206
STRUCK-SegTrack	**0.477**	**41.9**	**0.164**
Improvement	2.44%	14.27%	20.32%

3.6.3 Subjective Results

Figure 3.14 shows examples of Deer, Car1, Shaking, Couple, Bird1, Diving, Tiger2, Jumping, Skating2, Bolt2, and Football sequences and explores how the proposed method applied to all trackers (continuous boxes) reduces the tracking drift and successfully tracks the target object while the corresponding original trackers (dashed boxes) drift away from the target object.





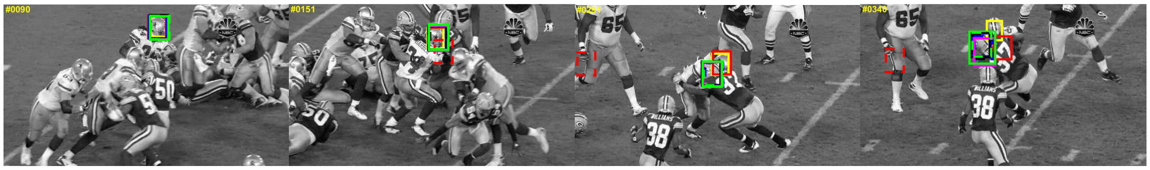


Figure 3.14: Subjective results using the proposed drift detection and correction.

As shown in Figure 3.14, different base trackers bounding boxes (KCF, SAMF, DSST, and STAPLE in the first video, KCF and STRUCK in second video, KCF, SAMF, and STAPLE in third video, KCF, and DSST in fourth video, STRUCK, KCF, and DSST in fifth video, STRUCK, SAMF, DSST, and STAPLE in sixth video) are drifted from the target object, while the proposed method successfully tracks the objects of interest for different challenges.

Low Resolution: The Deer sequence is used to evaluate the effectiveness of the trackers in dealing with scenarios of low contrast between the target and background. As shown in Figure 3.14 (1^{st} row), the proposed method allows all tested trackers to successfully track the target object. On the contrary, all base trackers, except of STRUCK tracker, are get distracted and drifted away from the target object.

Scale Variation: Car4 sequence is used to investigate the stability of the proposed method under the scale change challenge. Figure 3.14 (2^{nd} row) shows that the proposed method successfully tracks the target object without drift for all tested trackers, while STRUCK and KCF base trackers drift out of the target object at the middle of the sequence.

Illumination Variation: To study the effect of the illumination variation on the tracking quality, Shaking video sequence is used. Figure 3.14 (3^{rd} row) shows that the proposed method allows all tested trackers to effectively locate the target object, while KCF, SAMF, and STAPLE base trackers have lost the target after the drastic illumination change, and only SAMF could recover at the middle of the sequence.

Object Deformation: Figure 3.14 (4^{st} row) clarifies the effectiveness of the proposed method in handling object deformation. All tested trackers succeeded in handling the drastic deformation in the object under test, while KCF and DSST base trackers gradually drifted away from the target object at the beginning of the sequence.

Out-of-View: As shown in Figure 3.14 (5^{th} row), both KCF and DSST trackers drifted off the target of the Bird 1 sequence when it is out-of-view with no recovery, while all modified trackers have succeeded to track the target object without drift.

In-plane Rotation: For the Diving sequence, Figure 3.14 (6^{th} row), both SAMF and STAPLE base trackers drifted off the target after the beginning of the sequence until the end. On the other hand, all modified trackers tracked the target object without drift. While the base STRUCK tracker can track the target object successfully, the proposed method allows the modified STRUCK to locate more accurate BB around the target object.

Occlusion: The seventh row of Figure 3.14, for Tiger 2 sequence, shows that all base trackers and their corresponding modified versions successfully located the target object under occlusion.

Motion Blur: For the Jumping sequence, Figure 3.14 (8^{th} row), KCF, DSST, and STAPLE base trackers have big drifts as a result of motion blur in the image scene. However, the modified KCF, SAMF, and STAPLE trackers effectively locate and track the target object. The modified DSST tracker shows slightly better estimated BB than the base one. Both base and modified STRUCK trackers show successful tracking of the target object through the sequence.

Fast Motion: For the Skating 2 sequence, Figure 3.14 (9^{th} row), all base trackers are distracted by the fast motion of the target object. However, the proposed method allows the modified trackers to estimate bounding boxes that are closer to the ground-truth than the base trackers, especially STRUCK, DSST, and STAPLE trackers that drift off the target at the middle of the sequence. In addition to fast motion, the similar object near the target distracts most of the trackers to some extent.

Background Clutters: The tenth row of Figure 3.14, for Bolt 2 sequence, shows that KCF, SAMF, and DSST base trackers drift off the target object at the beginning of the sequence and their corresponding modified versions could successfully locate the target object in spite of high background clutters.

Out-of-Plane Rotation: Figure 3.14 (11^{th} row) shows the Football sequence in which STRUCK base tracker gradually drifts off the target at the beginning of the sequence and lost it at the end, while the modified version kept tracking the target object. The out-of-plane rotation challenge distracts the trackers as the target size changes. The modified STRUCK shows successful tracking, while other trackers (both base and modified) partially locate the target at the end of the sequence after successful tracking.

3.6.4 Proposed vs. Segmentation-based Tracking Methods

[74] tested their segmentation-based tracker (JOTS) using two data sets $SegTrack_{v1}$ [97] and $SegTrack_{v2}$ [98], and reported the average pixel error per frame for $SegTrack_{v1}$ dataset and the intersection-over-union overlap metric is reported for $SegTrack_{v2}$. These experiments demonstrate that JOTS performs favorably against state-of-the-art methods. However, authors manually selected different parameters for individual videos to achieve the best tracking quality assuming a user annotation followed by the interactive segmentation is provided in the first frame.

For fair comparison with our automated method on the publicly available OTB benchmark [56], we experimented with a common parameter set for experiments over all videos of both $SegTrack_{v1}$ and $SegTrack_{v2}$ data sets. On average, we achieved better quality than that achieved using the selected authors parameters for individual videos. The common parameters are labeling ratio that controls the size of the surrounding region of the target object (LR) and the outliers distance threshold that controls the maximum distance for outliers (TOD). For simulation, both LR and TOD are selected as 0.76 and 25 respectively. In addition, we tested an automatic segmentation of the target object in the original BB at the first frame using two approaches. The first approach uses OTSU threshold to segment the target object from the background. In the second approach, we used our automated seeded segmentation in section 3.5.2 to segment the target object. Experiments on both data sets showed that our seeded segmentation has achieved better tracking quality according to the two metrics suggested by authors of JOTS.

For comparison, the original KCF tracker, the proposed KCF tracker (KCF-SegTrack), and JOTS are compared over 35 videos from OTB benchmark. Table

3.7 shows that our tracker outperforms JOTS according to AOR, CLE, and FR measures. In addition, our KCF-SegTrack has clearly better tracking performance (28.04 FPS) compared with JOTS tracker (0.074 FPS).

Table 3.7: Comparison of proposed method vs. JOTS.

	AOR			**CLE**			**FR**		
	KCF	KCF-SegTrack	JOTS	KCF	KCF-SegTrack	JOTS	KCF	KCF-SegTrack	JOTS
Deer	0.62	**0.69**	0.165	21.39	**9.22**	97.77	0.154	**0.014**	0.636
Crowds	0.79	**0.80**	0.18	3.06	**2.99**	31.07	0	0	0.5
Couple	0.20	**0.35**	0.156	47.55	**22.33**	50.58	0.664	0.357	**0.121**
ClifBar	0.25	**0.40**	0.247	36.72	**14.871**	62.15	0.463	**0.004**	0.471
CarScale	0.41	0.43	**0.73**	**16.14**	14.11	13.4	0	0	0
CarDark	0.61	0.61	0.145	6.04	6.04	20.55	0	0	0
Car1	0.13	**0.17**	0.093	39.71	**2.17**	98.94	0.238	**0**	0.74
Box	0.30	0.28	**0.431**	89.12	121.316	**42.44**	0.431	0.543	**0.216**
BlurOwl	0.19	0.19	**0.238**	183.42	**79.31**	191.66	0.763	**0.424**	0.669
BlurFace	0.83	**0.85**	0.527	5.57	**4.56**	31.47	0	**0**	0.062
BlurBody	0.67	**0.67**	0.454	14.98	**14.98**	41.81	0.035	**0.035**	0.084
Bird1	0.05	**0.24**	0.16	194.82	74.19	**24.18**	0.879	0.504	**0.039**
Basketball	0.67	**0.68**	0.595	7.88	**7.88**	16.03	0	0	0
Biker	0.24	0.24	**0.621**	77.17	82	**5.62**	0.535	0.542	**0.012**
Trellis	0.63	0.63	0.395	7.76	**6.99**	20.13	0	0	0
Girl	0.54	0.54	0.161	11.48	**11.25**	30.89	0.092	0.075	**0**
Car4	0.48	0.48	0.322	9.87	**9.57**	86.7	0	**0**	0.372
Walking2	0.39	**0.45**	0.309	28.98	**23.18**	42.69	0	**0**	0.463
David	0.53	0.53	0.263	**8.06**	9.03	37.51	0	**0**	0.22
Sylvester	**0.64**	0.63	0.305	**12.91**	13.27	15.03	0.076	**0.076**	0.304
Shaking	0.03	**0.66**	0.102	112.5	**10.5**	20.41	0.835	0	0
BlurCar2	0.76	**0.74**	0.299	5.917	**7.378**	75.438	0	0	0.051
Diving	0.32	**0.33**	0.192	39.524	**26.377**	62.53	0.172	**0.051**	0.255
Freeman4	0.18	**0.40**	0.458	27.11	8	**7.36**	0.452	**0.053**	0.25
Human3	0.01	**0.22**	0.107	260.18	**98.13**	109.8	0.987	**0.541**	0.723
Dudek	0.73	**0.73**	0.185	20.03	**11.42**	155.75	0	**0**	0.305
Football	0.55	**0.57**	0.365	14.6	13.93	**13.32**	0.08	**0.06**	0.08
Human9	0.39	**0.39**	0.146	14.76	**13.73**	25.039	0	0	0.083
Human4	**0.37**	0.35	0.256	**131.7**	138.2	135.8	0.46	0.46	**0.45**
DragonBaby	0.31	0.31	0.224	50.39	50.39	55.85	0.3	0.3	0.327
Average	0.43	**0.4852**	0.29437	49.978	**30.243733**	54.064	0.2539	**0.1346333**	0.24777

3.6.5 Computational Costs

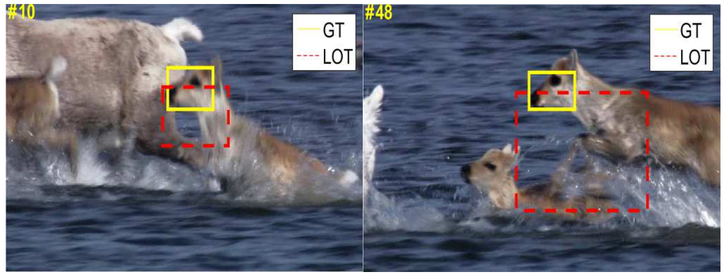
The computational burden for detecting and correcting the tracking drift remains a significant goal in order not to sacrifice the tracking system performance. In Table 3.8, we show the frame rate (the actual average number of frames processed per second) of each base tracker compared with the modified version using our proposed method over 100 test videos. In addition, we included the frame rate of all individual components of our method including seed selection and segmentation $SS - Seg$, drift detection DD, and drift correction DC. As shown, the $SS - Seg$ (implemented in C++) has relatively higher frame rate for STRUCK (implemented in C++) than other trackers (implemented in Matlab and use Mex files for seed selection and segmentation). The table also shows that for modified STRUCK tracker, the frame rate is lower compared with its individual components since integrating the segmentation with the tracking for all test sequences consumes more memory resources that affects the overall performance. The DD has high frame rate which shows that using saliency feature slightly affects the tracking performance. Finally, the DC has achieved very high frame rate as it considers only the relocation of the tracker position.

Table 3.8: Frame rate of the proposed drift detection and correction method.

Trackers	Base Tracker	Modified Tracker (ours)	Individual Components			FPS % of Reduction	Impl.
			SS-Seg	DD	DC		
STRUCK	25.09	18.04	56.1	318.6	5263	28%	C++
KCF	55.3	34.7	47.62	138.2	242891	37%	Matlab & Mex
SAMF	19.08	14.9	27.7	88.96	189050	22%	Matlab & Mex
DSST	31.49	24.82	37.65	240.72	1057	21%	Matlab & Mex
STAPLE	47.78	28.84	31.08	224.18	72775	39%	Matlab & Mex

3.6.6 Advantages and Limitations

The modified trackers generate a BB that is better-placed (in terms of objective measures) than the BB of the base trackers. This is because we 1) restrict object seeds to be always inside the original output BB; 2) adapt the seeds to the BB itself; and 3) use a powerful segmentation approach [75]. Our method achieves better improvement when the output BB deviates *reasonably*, i.e., not too much, from its true position. When the original BB deviation is high, (e.g., when off-target as in Figure 3.15(e)), or when the size of BB is wrongly estimated as in Figure 3.15(b-d), the improvement is small. For example, deviations of the LOT tracker [12] output BB are high; Figure 3.15; and thus the improvement using the proposed method is small (AOR at 1.29% and CLE at 1.80%). Our proposed method has lower FPS as shown in Table 3.8; the use of fast G-Cut segmentation methods [78, 99–101] can increase the overall FPS.



(a) Abrupt change. (b) Too large BB.

(c) Small on-target. (d) Small off-target.

Figure 3.15: Base tracker related limitations.

Our drift detection uses the saliency information of the target inside the B_t to detect if it starts to drift, then it relocates the B_t around the target object by applying segmentation of that object. Experiments show that the base trackers may have random behavior at certain frames that causes the tracker to suddenly drift off the target immediately in next frame. As a result, the proposed tracker may wrongly switch to another object that might have different segmentation and saliency characteristics. Such behavior may mislead the proposed approach.

3.7 Conclusion

This chapter proposed a method for drift detection, using saliency features of the target objects, integrated with a drift correction mechanism through seeded segmentation of the estimated tracking output bounding box to improve the tracking quality. Instead of applying segmentation at each frame, the proposed method applies segmentation only at the occurrence of tracking drift. Our tracker-independent method is applied to five recent different performing trackers on a large publicly available data set of 100 test sequences. Results indicate that our method reduces the tracking drifts that in turn leads to an improved overall tracking quality of all tested trackers according to various evaluation criteria.

As future work, it is intended to study the effect of using different saliency detection algorithms for drift detection and to study the effect of using fast segmentation on the tracking performance. In addition, it is desirable to integrate both saliency features and objectness measures in a unified framework for automatic drift detection.

Chapter 4

Artificial Immune System Based Parameter Optimization of SVM in Object Tracking

4.1 Abstract

Little research attention has been given to study the use of artificial immune system (AIS) optimization in computer vision. Support vector machine (SVM) classification approach has been widely and successfully used for diverse applications including object tracking and segmentation. Establishing an efficient SVM model requires careful selection of its penalize and kernel parameters that have strong influence on the classification accuracy. Long and tedious trial and error approaches are usually used for the selection of SVM parameters for a specific application, and these approaches do not guarantee best performing parameters. This chapter proposes a method for adaptive selection of SVM parameters for enhanced object tracking using AIS optimization. This method incorporates a complementary SVM model, that is trained on-line, in which the AIS is used to automatically select the near-optimal parameter set for the tracking model without human intervention. The proposed method is tested on the STRUCK tracking algorithm which uses SVM [2]. The obtained results show that the performance

of the proposed approach well outperforms the original STRUCK.

4.2 List of Symbols

Symbol	Description
S_t^+	Positive sample patches
S_t^-	Negative sample patches
v_t^+	Features of positive sample patches
v_t^-	Features of negative sample patches
N_p	Number of positive samples
N_n	Number of negative samples
c_v	Feature variation constant
N_α	Number of correctly classified data in SVM model
σ_r	Range of kernel parameter σ
C_r	Range of penalize parameter C
P_s	AIS population size
A_s	AIS Archive size
K	Number of times classification is performed
K_{max}	AIS maximum number of iterations
$\{C_t, \sigma_t\}$	Optimal SVM parameter set calculated by AIS model
$v_{\Delta t}^+$	Variation of positive features
$v_{\Delta t}^-$	Variation of negative features
c_a	Tolerance to terminate the AIS optimization process

4.3 Introduction

Object tracking has many uses [4, 102] such as surveillance, robotics, and augmented reality. Many powerful approaches treat the tracking problem as a classification task and use on-line learning techniques to update the object model [2, 9, 103–106]. A structured output learning with SVM enables adaptive tracking [2, 104, 106]. SVM, as a supervised learning method, constructs a classification model using training data [107]. SVM uses the kernel function to map the input data into a high-dimensional feature space and find a near-optimal hyperplane for classification. Different kernel functions can be used to select such support vectors along the surface of the hyperplane including linear, polynomial, sigmoid and radial basis function. Although SVM has been successfully applied in many fields such as object recognition [108,109], and tracking [2,17,110], there is a conspicuous problem in the practical application of SVM. Selection of appropriate kernel function and its parameters affects learning and generalization performance of SVM and is important to obtain the best classification performance that in turn leads to minimal prediction error.

This chapter proposes to use AIS-based parameter selection of SVM algorithm. For SVM-based object tracking such as STRUCK [2], we use AIS for the selection of near-optimal kernel and penalize parameters according to the selected features of the tracking model. The proposed method incorporates a complementary SVM model that is trained on-line using the features of the current frame in which the classification accuracy is considered as an evaluation function.

The rest of the chapter is organized as follows. Section 4.4 presents prior work and its relation to the proposed approach. Section 4.5 introduces the proposed approach for enhanced SVM-based object tracking using AIS. The experimental results are presented in Section 4.6, followed by a conclusion in Section 4.7.

4.4 Prior Work

Powerful tracking approaches use on-line learning techniques to update the object model. In [17], S. Hare et al. presented a framework for adaptive visual object tracking (STRUCK) based on structured output prediction using a kernelized SVM. In their work, the authors predefined the SVM model with fixed parameters regardless of the varying nature of video signals. In [110], Zhang et al. proposed a multi-view learning framework using multiple SVMs, based on multiple views of features and a novel combination strategy. For comprehensive representation, the authors selected three different types of features to train the corresponding SVMs. However, the fixed parameter set used by such approach may affect the tracking accuracy due to the varying nature of video signals from which the features are extracted. In [111], Keerthi proposed an algorithm to find the optimal parameters of the least-squares SVM by gradient descent method. However, gradient descent is sensitive to initial parameters and may converge to local optimum if initialization is far from the optimal solution. Lessmann et al. [112]

optimized the SVM parameters by two genetic algorithm methods. Compared with genetic algorithms, AIS can effectively avoid the premature convergence and guarantee the variety of solution [113].

Numerous researches have applied AIS to improve the performance of classification algorithms such as SVMs [14, 15, 61, 114]. But, all of these use static and pre-defined parameters for SVM. They do not select the SVM parameters on-line, meaning adaptive to the current state of the system. In [114], Aydin et al. proposed an AIS-based SVM algorithm for fault diagnosis of induction motors. In their method, the number of the miss-classified data is considered as an evaluation function and a radial basis kernel is used. An off-line training stage is adopted to find the optimal SVM parameters to be used for testing. Lin and Chen [14] applied the AIS algorithm to enhance the classifying capacity of the case-based reasoning algorithm. In [15], Aydin et al. proposed a multi-objective AIS to optimize the radial based kernel and penalize parameters of SVM. In their method, a training stage of SVM is done off-line in which multiple solutions are found by using AIS model and then these parameters are evaluated in a test stage. In [61], Wu et al. proposed an AIS-based SVM approach for classifying ultrasound breast tumor images. Their method adopts both parameter tuning and feature selection to achieve higher lesion classification accuracy. The results showed that some lesions still have similar characteristics that cause the CAD system to fail. Moreover, an off-line training is used in order to find the best performing parameters.

The above-mentioned approaches apply the AIS using pre-defined features prior to the testing phase, which is impractical for object tracking due to the varying nature of video signals. Accordingly, an on-line parameter selection framework is important. In [115], Chau et al., training video sequences are classified off-line according to their contextual features and then once a context change is detected, the tracking parameters are tuned using the learned values. However, it is impractical for automated object tracking. The method learns how to tune the tracker parameters to cope with the tracking context (set of features) variations.

To the best knowledge of the author of this work, no method exists that selects the SVM parameters fully on-line, that is adaptive to the state of the system. Current

AIS approaches for SVM optimization [14, 15, 61, 114] apply the training off-line with pre-determined features and then use the resulting parameters for testing. Our proposed method applies the training of SVM in an on-line manner, that is when a variation in the features between consecutive frames is detected, with no restriction about the feature type used for training. The proposed method automatically selects a configuration that is best performing according to this variation in newly captured features between consecutive frames.

4.5 Proposed AIS Approach for SVM Optimization

4.5.1 Background

In object tracking, SVM-based classification aims to learn margin-based discriminative classifiers for maximizing the separability between object and background. Kernel selection and efficient kernel computation are important for robust tracking. STRUCK [2, 17] directly links the tracking to learning; it learns a prediction function f to directly estimate the new object position $\hat{y}$ between frames based on the feature vector x of y. It learns f in a structured output SVM framework, that uses a discriminant (or scoring) function F for prediction as $\hat{y} = f(x) = \arg\max_y F(x, y)$. Thus it performs a maximisation so to predict the new object position, and the discriminant function F includes y explicitly in the learning algorithm. To update the prediction function online, STRUCK supplies a labelled example y relative to the new tracker location. STRUCK consists of the following main steps: estimate change in target position, update discriminant function, Sequential Minimal Optimization (SMO), and budgeting. STRUCK learns discriminatively a scoring function F over input-output example set $\{(x_i, y_i)\}$. F maps the output BB y and its corresponding feature x to a scalar label. Once F is learned, the prediction of the output $\hat{y}$ highest compatible with the input x is obtained by maximizing F over all possible outputs y as $\hat{y} = \arg\max_y F(x, y)$.

The scoring function is in the form $F(x, y) = \langle w, \mathbf{\Phi}(x, y)\rangle$, where the weight vector w is learned with sequentially obtained example pairs in the set of training examples, $\mathbf{\Phi}(x, y)$ is a joint feature map, and $\langle \cdot, \cdot \rangle$ is the inner product. The scoring function

F can be learned by minimizing a constrained convex objective function subject to $\langle w, \delta\mathbf{\Phi}_i(y)\rangle \geq \Delta(y_i, y) - \xi_i$ with the slack variables ξ_i allowing the examples to violate the constraint of being outside of the margin, $\delta\mathbf{\Phi}_i(y) = \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, y)$. Instead of solving the primal optimization using a convex objective function, its dual formulation using the Lagrangian function is obtained as given in (4.1), where the Lagrangian multiplier α corresponds to the margin constraint $\delta\mathbf{\Phi}_i(\mathbf{y})$ in the convex objective function. In SVM, it is crucial to carefully design the kernel function $k(x, y, \bar{x}, \bar{y}) = \langle\mathbf{\Phi}(x, y), \mathbf{\Phi}(\bar{x}, \bar{y})\rangle$ for the optimization problem.

When equipped with kernel functions, SVM learning algorithms encounter a problem of dimensionality that causes unbounded linear growth in model size and update time with the amount of training data. Therefore, it is important to upper bound the number of support vectors that are generated during tracking. STRUCK uses an on-line budgeting mechanism to solve such problem. During tracking, a pool of support vectors S is maintained after the on-line budgeting process in each frame. During SMO, for each sample x_i with corresponding BB y in S, the coefficient α_i^y is incrementally updated. The only support vectors that correspond to non-zero α_i^y are kept in pool S. As an advantage of budgeting mechanism, given a new training example (x_i, y_i), the algorithm is optimized to maximize the margin of the SVM based on α_i, keeping the number of maintained support vectors at upper bound. Using standard Lagrangian duality techniques, STRUCK uses an objective function in dual form as

$$
\begin{aligned}
&\max_{\alpha_i^y}\left(\sum_{i,y\neq y_i}\triangle(y,y_i)\alpha_i^y - \frac{1}{2}\sum_{i,y\neq y_i,j,\bar{y}\neq y_j}\alpha_i^y\alpha_j^{\bar{y}}\langle\mathbf{\Phi}(x_i,y),\mathbf{\Phi}(x_j,\bar{y})\rangle\right) \\
&\text{s.t. } \alpha_i^y \geq 0, \forall i, \forall y \neq y_i \quad \text{and} \quad \sum_{y\neq y_i}\alpha_i^y \leq C, \ \forall i,
\end{aligned}
\tag{4.1}
$$

where $\triangle(y, y_i)$ is a loss function between the BB y of object of interest and the sample y_i, α_i^y is the α parameter of sample y_i with BB y, $\alpha_j^{\bar{y}}$ is the α parameter of sample j with BB $\bar{y}$, $k(x, y, \bar{x}, \bar{y}) = \langle\mathbf{\Phi}(x, y), \mathbf{\Phi}(\bar{x}, \bar{y})\rangle$ is the kernel similarity function which is defined as simple inner product between samples x_i of BB y and sample x_j of BB $\bar{y}$, and C is a penalize (regularization) parameter. Using the loss function $\triangle(y, y_i)$

between sample pair of BB y and y_i allows the discrimination function (that internally discriminates between the two classes of data) to treat a test sample y_i according to its closeness to the object of interest y instead of treating all samples equally. The SMO is an iterative method that jointly optimize pairs of α parameters that are chosen so as to maximize the objective function in (4.1). SMO adjusts the bias parameter that is used for the construction of the discrimination function of SVM. SMO repeats such steps until convergence. Results show that STRUCK is able to identify distinct appearances of the object over time. Moreover, the budgeting mechanism maintains support vectors from the entire tracking sequence and does not discard old appearance information which helps prevent drift during tracking.

In STRUCK [2,17], Hare et al. use a fixed budget size $C = 100$ and define the *feature responses* as a feature vector x of sample image patch y, and apply a Gaussian similarity kernel $k(x, \bar{x}) = exp(-\sigma \|x - \bar{x}\|^2)$ between the two pairs (x, y) and $(\bar{x}, \bar{y})$, with fixed $\sigma = 0.2$. The use of fixed SVM parameters C and σ does not consider the varying nature of video signals which affects the tracking quality. Accordingly, we propose to adaptively select the best performing penalize C and kernel σ parameters of STRUCK tracker using AIS optimization according to the current image features.

4.5.2 Our Approach

At a given frame F_t, the original STRUCK tracking model extracts different positive and negative sample patches around the estimated B_{t-1} in F_{t-1}, that incorporates a number N_p of positive samples and a number N_n of negative samples. In our approach, we extract the feature sets $\{v_t^+\}$ and $\{v_t^-\}$ corresponding to the positive and negative samples respectively, to train the SVM model (SVM_1) with *pre-defined* C and σ parameters to find the estimated object location. To update these parameters, we propose an adaptive parameter selection approach as shown in Figure 4.1. We use an AIS based optimization approach to estimate the near-optimal parameter set of the main SVM model (SVM_1), based on the training of an additional SVM (SVM_2). At frame F_t ($t \geq 1$), we train SVM_2 on-line using the extracted features of the already generated sample patches (in F_t), and we optimize its parameters by implementing a CLONALG algorithm. The obtained parameter set $\{C_t, \sigma_t\}$ is then used by SVM_1 of the tracking model at frame F_t.

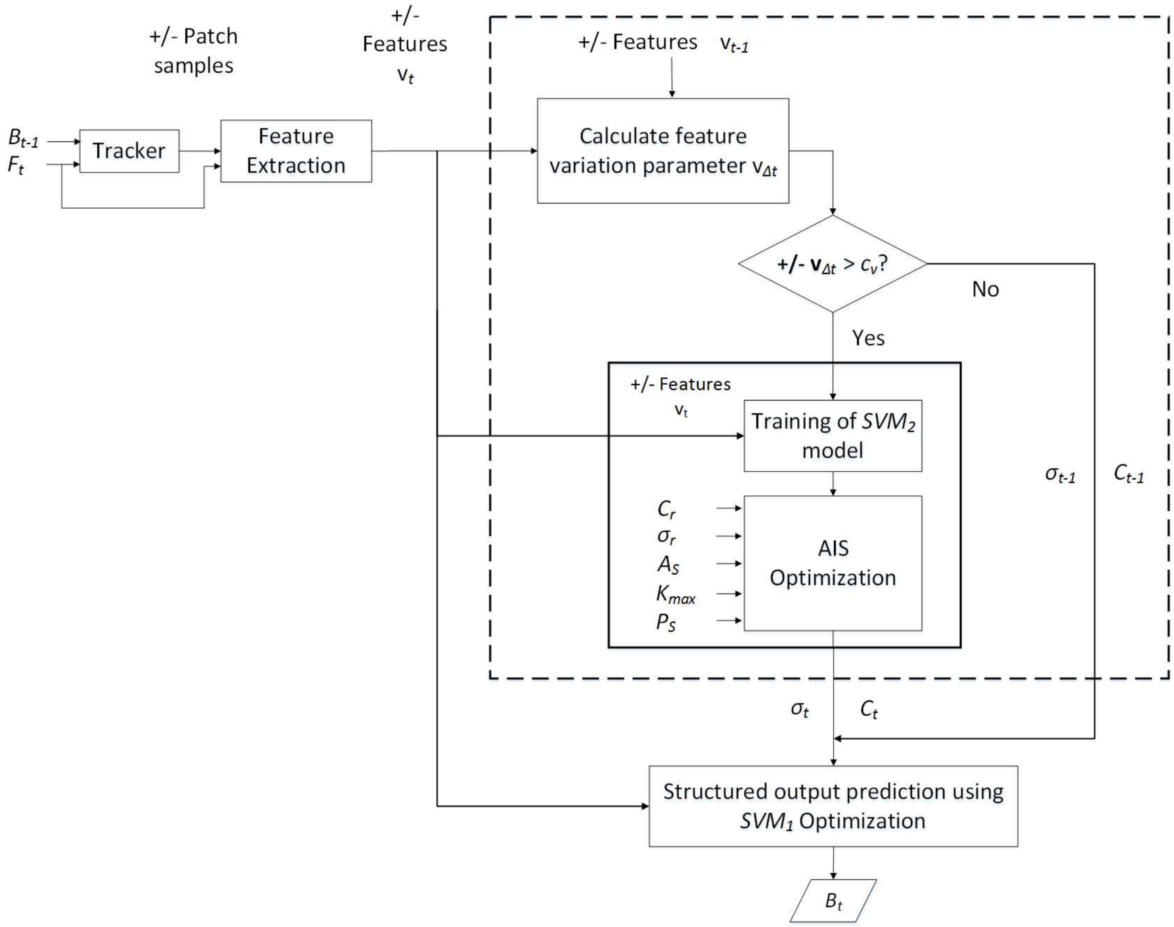


Figure 4.1: Block diagram of the proposed AIS method for SVM optimization.

We calculate the positive $v_{\Delta t}^{+}$ and negative $v_{\Delta t}^{-}$ feature variation parameters based on the similarity of the corresponding features between frames F_t and F_{t-1} as

$$v_{\Delta t}^{+} = 1 - Sim(\{v_t^{+}\}, \{v_{t-1}^{+}\}). \tag{4.2}$$

$$v_{\Delta t}^{-} = 1 - Sim(\{v_t^{-}\}, \{v_{t-1}^{-}\}), \tag{4.3}$$

where Sim is the similarity function that measures the ratio between the number of similar features and the number of all distinctive features of F_t and F_{t-1}, measured for positive and negative features separately. For simplicity, in (4.2) and (4.3), we use terms

$\{v_t^-\}$ and $\{v_{t-1}^-\}$ without indicating the number of elements in each set as it might be different between F_{t-1} and F_t. $v_{\Delta t}^-$ and $v_{\Delta t}^+$ are used to control the use of the AIS-based optimization as

$$vAIS = \begin{cases} 1 & : (v_{\Delta t}^+ > c_v) \vee (v_{\Delta t}^- > c_v) \\ 0 & : otherwise, \end{cases} \tag{4.4}$$

where c_v is a constant for the acceptable variation in either positive or negative features. Depending on the extracted features $\{v_t^+\}$ and $\{v_t^-\}$ we train SVM_2 for AIS to calculate the best parameter set for SVM_1 of the tracking model at current frame F_t. For AIS optimization, the corresponding range C_r and σ_r of C and σ parameters, the maximum number of iterations K_{max}, population size P_S, and the archive size A_S are required. Our objective function here is to maximize the classification accuracy of the trained SVM_2 model as

$$\begin{aligned} \max_{\{\alpha_k\}} \quad & \frac{\left(\sum_{k=1; \alpha_k \in \{v_t^+ \cup v_t^-\}}^{K} N_{\alpha_k}(C_t, \sigma_t)/M \right)}{K} \\ \text{s. t.} \quad & \sigma_t \in \sigma_r, C_t \in C_r, \end{aligned} \tag{4.5}$$

where $N_{\alpha_k}(C_t, \sigma_t)$ is the number of correctly classified data in SVM_2 trained model according to (C_t, σ_t) candidate antibody, $\{\alpha_k\}$ is a set of M randomly selected features from the set of all features extracted in F_t, and K represents the number of times the classification is performed. The accuracy of SVM_2 is calculated K times and the average of the K tests is to be maximized to avoid the randomness effect. If no feature variation, no AIS is used and the tracking continues with the $\{C_{t-1}, \sigma_{t-1}\}$ parameter set.

Algorithm 4.1 details the steps of the proposed method. We start the AIS optimization process by setting the model parameters: the population size P_S, the archive size A_S, the maximum iterations K_{max}, and parameters ranges C_r, and σ_r. We set c_a a constant to decide if the accuracy obtained is good enough to terminate the optimization process and c_v a constant to decide if change in the features is significant.

At frame F_t, we then measure the similarity of features between F_t and F_{t-1} (both positive and negative features are considered separately) in order to calculate the feature variation $v^+_{\Delta t}$ (for positive features) and $v^-_{\Delta t}$ (for negative features). If either $v^+_{\Delta t}$ or $v^-_{\Delta t}$ is above c_v, then SVM_2 is trained on-line using both positive $\{v^+_t\}$ and negative features $\{v^-_t\}$ extracted from F_t. The AIS uses the trained SVM_2 model (SVM_{2tr}) to select the near-optimal parameter set $\{C_t, \sigma_t\}$ at F_t that maximizes the classification accuracy of SVM_2 according to (4.5). This parameter set is used by SVM_1 tracking model for the F_t. AIS randomly generates a population of P_S antibodies ab as pairs of (C, σ) according to the selected ranges C_r, and σ_r of both kernel and penalize parameters, respectively. Such population is considered as an initial generation. After the initial antibody population is produced, SVM_2 is trained for each antibody by SMO. In this step, our objective function is the classification accuracy as defined in (4.5) and we find the affinity set $\{a_i\}$ of all P_S antibodies and antigens. In line 9 of Algorithm 4.1, we evaluate for each candidate antibody if it can be selected as one of the problem candidate solutions according to the given objective function (4.5). We have many candidate solutions and the evaluation scores (affinity) are stored in $\{a_i\}$ where i represents the antibody (candidate solution) i. Here, our goal is to select the C and σ that maximize the classification accuracy of the SVM model. We evaluate the objective function for all candidate antibodies. The dominance of all antibodies is checked to obtain the best (C, σ) pair that best matches the objective function in (4.5). The non-dominated antibody ab (the ab that perfectly match the objective function) are selected as the best ab to up-date the archive of best solutions. The cloning process clones the non-dominated solution as well as the dominated ones in order to ensure the diversity in the optimization. Then, each ab is encoded as a binary string and the mutation process is applied. At the end of each generation, the archive is updated with the best solutions. After K_{max} iterations or if the affinity a_i of an antibody exceeds $c_a = 0.95$, the algorithm terminates and the best parameter set is found in the archive.

Algorithm 4.1: AIS-based SVM parameter optimization.

Data: Population size P_S; Archive size A_S; Maximum iterations K_{max}; Parameters ranges C_r and σ_r; constants c_a and c_v

Result: Optimal parameters C_t and σ_t at frame F_t

1 $vAIS = 0$;
2 **for** *frame* F_t **do**
3     $v^+_{\Delta t} = 1 - Sim(\{v^+_t\}, \{v^+_{t-1}\})$;
4     $v^-_{\Delta t} = 1 - Sim(\{v^-_t\}, \{v^-_{t-1}\})$;
5   **if** $(v^+_{\Delta t} > c_v \vee v^-_{\Delta t} > c_v)$
6         $SVM_{2tr} = Train(SVM_2, \{v^+_t\}, \{v^-_t\})$;
7   History = initial population $\{C_i \ , \sigma_i\}$; i=1, ..., P_S;
8   **for** *k = 1 to* K_{max} **do**
9             $\{a_i\}$ = Evaluate($SVM_{2tr}, \{v^+_t\}, \{v^-_t\}, C_i, \sigma_i$) ;
10   Dominance ($\{a_i\}$);
11             $\{C_i, \sigma_i\}_{clone}$ = Clone ($\{C_i, \sigma_i\}$) ;
12   Archive = Update-Archive ($\{C_i, \sigma_i\}_{clone}$) ;
13             $\{C_i, \sigma_i\}_{binary}$ = Encoding ($\{C_i, \sigma_i\}_{clone}$) ;
14             $\{C_i, \sigma_i\}_{mutate}$ = Mutation ($\{C_i, \sigma_i\}_{binary}$);
15   History = Update-History ($\{C_i, \sigma_i\}_{mutate}$);
16   **if** $(Max(\{a_i\}) > c_a)$
17   break ;
18   **end**
19             $k = k + 1$;
20         $\{C_t \ , \sigma_t\}$ = Max (Archive);
21   STRUCK($F_t, SVM_1, C_t, \sigma_t, \{v^+_t\}, \{v^-_t\}$);
22   else
23   STRUCK($F_t, SVM_1, C_{t-1}, \sigma_{t-1}, \{v^+_t\}, \{v^-_t\}$);
24   **end**;

4.6 Results

4.6.1 AIS Model Simulation

To examine the convergence to the near-optimal solution, we test CLONALG-based AIS model using two functions: *Test* and *Rastrigin's* [116]. These functions are used as objective functions to evaluate solutions (see line 9 of Algorithm 4.1). To test if the proposed AIS model can successfully find the near global optimal of certain problem, we use a function (e.g., *Test*) with known global maximum/minimum and check if our model can find these. For this, we first run the AIS model after defining its objective function to be the maximizing of the *Test* function. To verify, we apply then the same on the *Rastrigin* function. In both cases, our AIS model was able to confirm the maxima/minima. For speed considerations (performance), we changed the parameters (population size, archive size, maximum iterations, parameters ranges) of the AIS model for each of the *Test* and *Rastrigin* functions and we checked the convergence rate of the solution (how quickly the AIS model finds the near optimal solution) and then, we choose those parameters that allow the model to find the near global solution with fast convergence rate (quickly find the solution in least iteration), see Figure 4.3.

The function *Test* is defined as

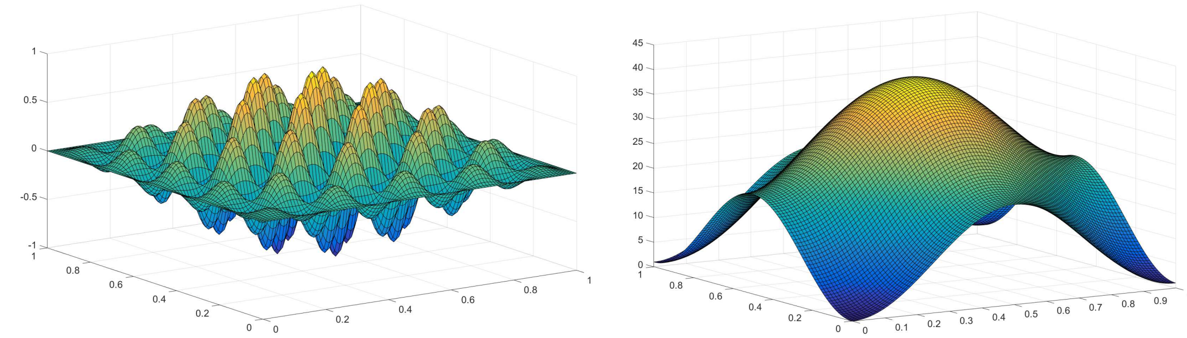
$$Test(x, y) = (15xy)(1 - x)(1 - y)sin(9\pi x)sin(9\pi y). \tag{4.6}$$

As shown in Figure 4.2a, the function *Test* has many local minimums and maximums, and is evaluated by generating numbers in the range from 0 to 1 in steps of 0.01 for the two dimensions. The global maximum and minimum values for this function are 0.9375 and -0.89168, respectively. The *Rastrigin* function, Figure 4.2 (b), is a non-convex function used as a performance test problem for optimization algorithms. Finding the minimum of this function is a fairly difficult problem due to its large search space and its large number of local minima. It is defined on an n-dimensional domain as

$$Rastrigin(x, y) = 10n + \sum_{i=1}^{n}[x_i^2 - 10cos(2\pi x_i)] \tag{4.7}$$

The test area is restricted to hypercube $0 \leq x_i \leq 1; i = 1, \ldots, n$. Its global minimum equal zero is obtainable for $x_i = 0, i = 1, \ldots, n$. As mentioned, the CLONALG-based AIS model converges successfully to the global maximum and minimum optimum solution for both *Test* and *Rastrigin* functions.

We tested the AIS model with both *Test* and *Rastrigin* functions with different AIS parameters and we selected those parameters that show near global solution with fast convergence rate for better performance as illustrated in Figure 4.3: using a population size of either $P_S = 20$ and $P_S = 30$ with archive size of $A_S = 4$ (dotted magenta



(a) *Test* function. (b) *Rastrigin* function.

Figure 4.2: Functions for the evaluation of the AIS model.

and black) leads to near global solution under the *Test* function. Other P_S values do not reach the near global solution. For our SVM/STRUCK simulations, we selected $P_S = 20$ as it achieves better performance with large feature vectors. Based on our simulations with *Test* and *Rastrigin*, we propose to use the AIS model parameters as default parameters in all our SVM/STRUCK simulations in section 4.6.3. Note that in Figures 4.3 and 4.4, "Fitness" refers to the output of the objective function evaluation in Algorithm 4.1.

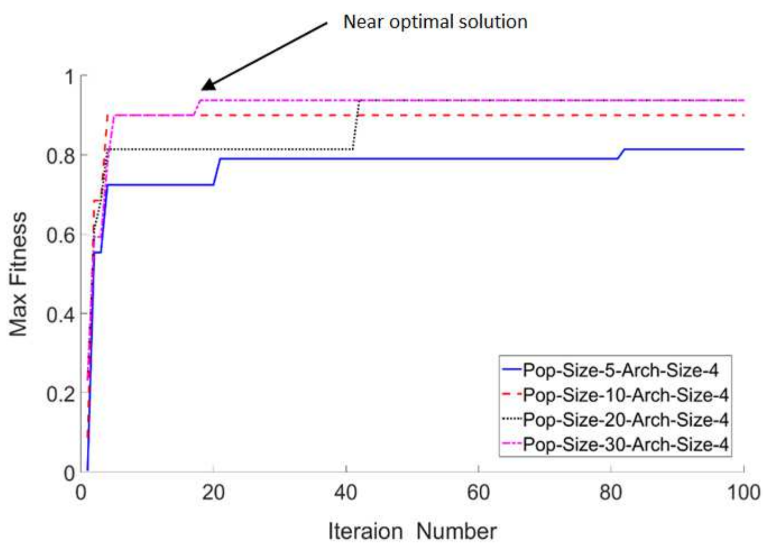


Figure 4.3: Convergence rate of *Test* function with different AIS parameters.

For the *Test* function in (4.6), Figure 4.4 shows the search range used by the CLONALG-based AIS model to find a near-optimal solution. The figure shows how the

AIS search space covers diverse of candidate solutions to find the near optimal solution. For each iteration (x axis), the fitness value obtained by the AIS model (z axis) is drawn for each population (y axis). We developed the model in a way so it can find the near global solution of different optimization problems given the objective function.

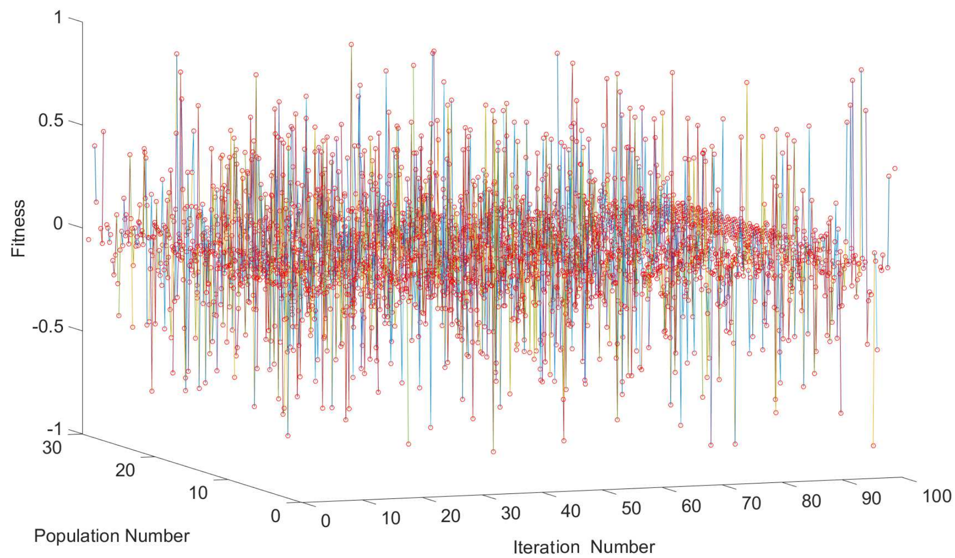


Figure 4.4: Search range of the CLONALG-based AIS model using *Test* function.

4.6.2 Experimental Setup

For experiments, we select the AIS model parameters that achieve both better accuracy and performance (see section 4.6.1) as follows: Population size $P_S = 20$; Archive size $A_S = 4$; Maximum iterations $K_{max} = 20$; Parameters ranges $C_r = [1, 255]$ and $\sigma_r = [0.01, 1]$. We set $c_v = 0.5$ as variation threshold and $c_a = 0.95$ as an acceptable accuracy to terminate the optimization process. For SVM_2, the classification accuracy is calculated five times on the trained model to obtain fair objective statistics.

For the training of SVM_1, 6 different types of Haar-like features are used. These features are arranged at 2 scales on a 4×4 grid, resulting in 192 features, where each feature is normalized to give a value in the range $[-1, 1]$ as used by STRUCK. For SVM_2, we used the same features in order to match the same information given by STRUCK without adopting external source of information. For original STRUCK, we used a budget size of 100, $C = 100$, and $\sigma = 0.2$.

We used a dataset of 35 video sequences that cover 11 tracking challenges [56]. We run each experiment three times on each sequence to obtain fair objective statistics. To assess tracking quality, we use: overlap ratio AOR, center location error CLE, failure rate FR. the success and precision plots of [56, 57]. Success plot represents the percentage of frames for which the overlap exceeds a certain threshold. Thresholds in the range [0, 1] with an increment step of 0.05 are used. The precision plot measures the percentage of frames in which the estimated locations are within a certain distance of the ground-truth positions. Such plot is measured in the range [0, 50] with an increment step of 5 pixels.

4.6.3 Objective and Subjective Tracking Improvement

Figures 4.5, 4.6, and 4.7 show the AOR, CLE, and FR plots of the improvement achieved by the proposed method over all test sequences. As can be seen, the quality of the modified STRUCK has achieved an average improvement of 20.0% (0.44 vs. 0.528), 54.5% (57.73 vs. 26.26), and 51.6% (0.248 vs. 0.12) for AOR, CLE, and FR, respectively. This shows that the proposed AIS-based model parameters have improved the classification accuracy, leading to an enhanced tracking quality. Concerning Precision and Success plots, Figures 4.8 and 4.9 show that the modified STRUCK tracker (blue) achieves better quality than the base STRUCK over a wide range of thresholds.

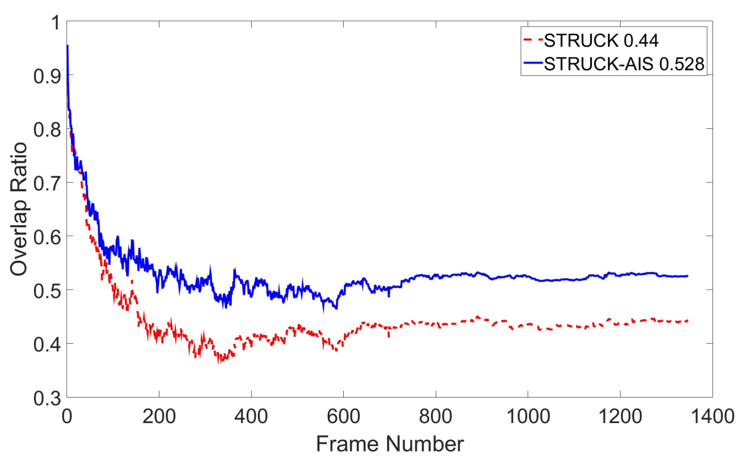


Figure 4.5: AOR plot of base and modified STRUCK trackers over all test videos.

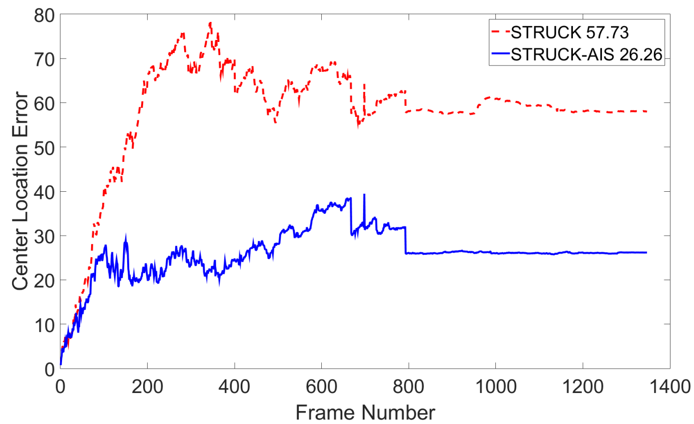


Figure 4.6: CLE plot of base and modified STRUCK trackers over all test videos.

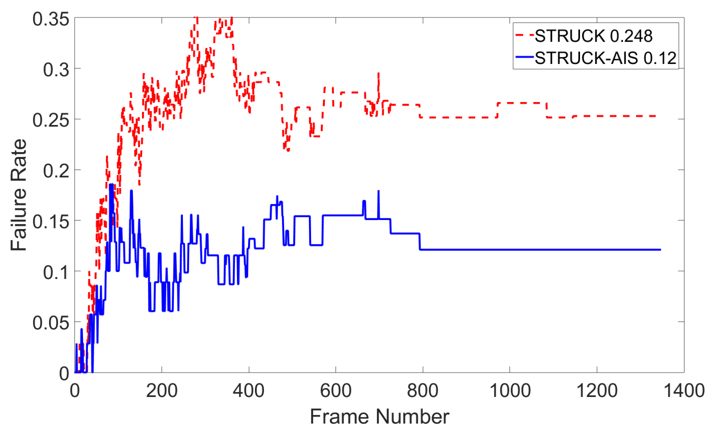


Figure 4.7: FR plot of base and modified STRUCK trackers over all test videos.

Table 4.1 shows the AOR, CLE, and FR quality measures of the modified STRUCK compared with the base STRUCK for individual test sequences; better results (higher AOR, lower CLE, or lower FR) are shown in bold and similar results are in italic. The modified STRUCK has significantly reduced the FR of the base STRUCK for challenging videos such as Board, David, Shaking, BlurFace Crowds, and Ironman. The CLE and FR metrics show better improvement than AOR metric due to the fact that STRUCK tracking algorithm is scale invariant and hence, it is expected that the AOR will achieve lower improvement, especially when the target object is far from the camera (small size) and the estimated BB is larger than its corresponding ground truth BB. Our proposed method has low frame rate (0.01 FPS) compared with the base tracking algorithm (26.82 FPS).

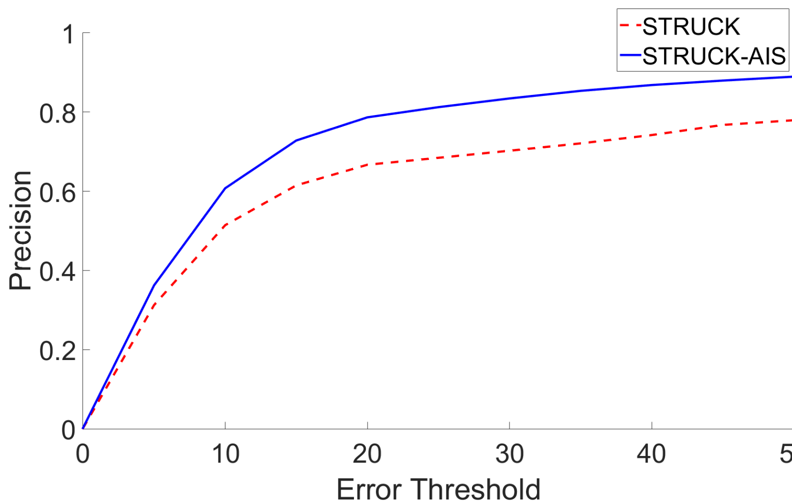


Figure 4.8: Precision plot of base and modified STRUCK trackers over all test sequences.

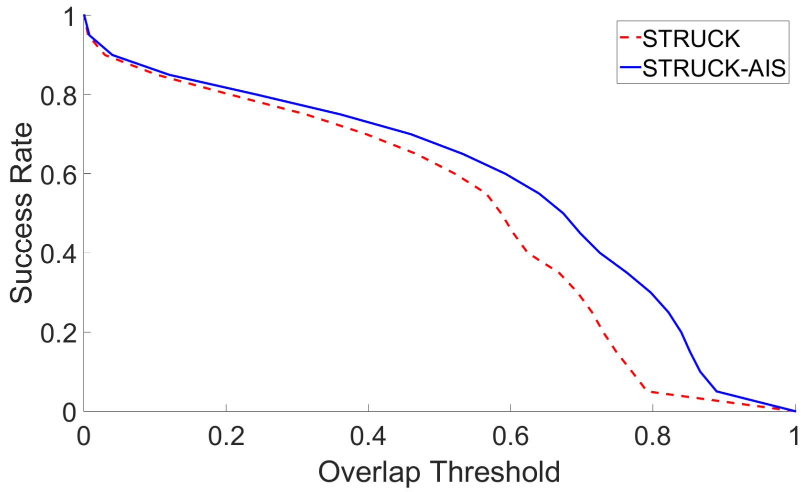


Figure 4.9: Success plot of base and modified STRUCK trackers over all test sequences.

Figure 4.10 shows subjective examples of improvement (STRUCK-AIS) using the proposed method applied to the STRUCK tracker [2] of different challenges: the modified STRUCK has better estimated BB (dotted blue) that are closer to that of the ground truth (yellow), while the base STRUCK (dashed red) drifts away from the target object. In addition, the modified STRUCK is shown to reduce the tracking drift that in turn leads to better tracking quality.

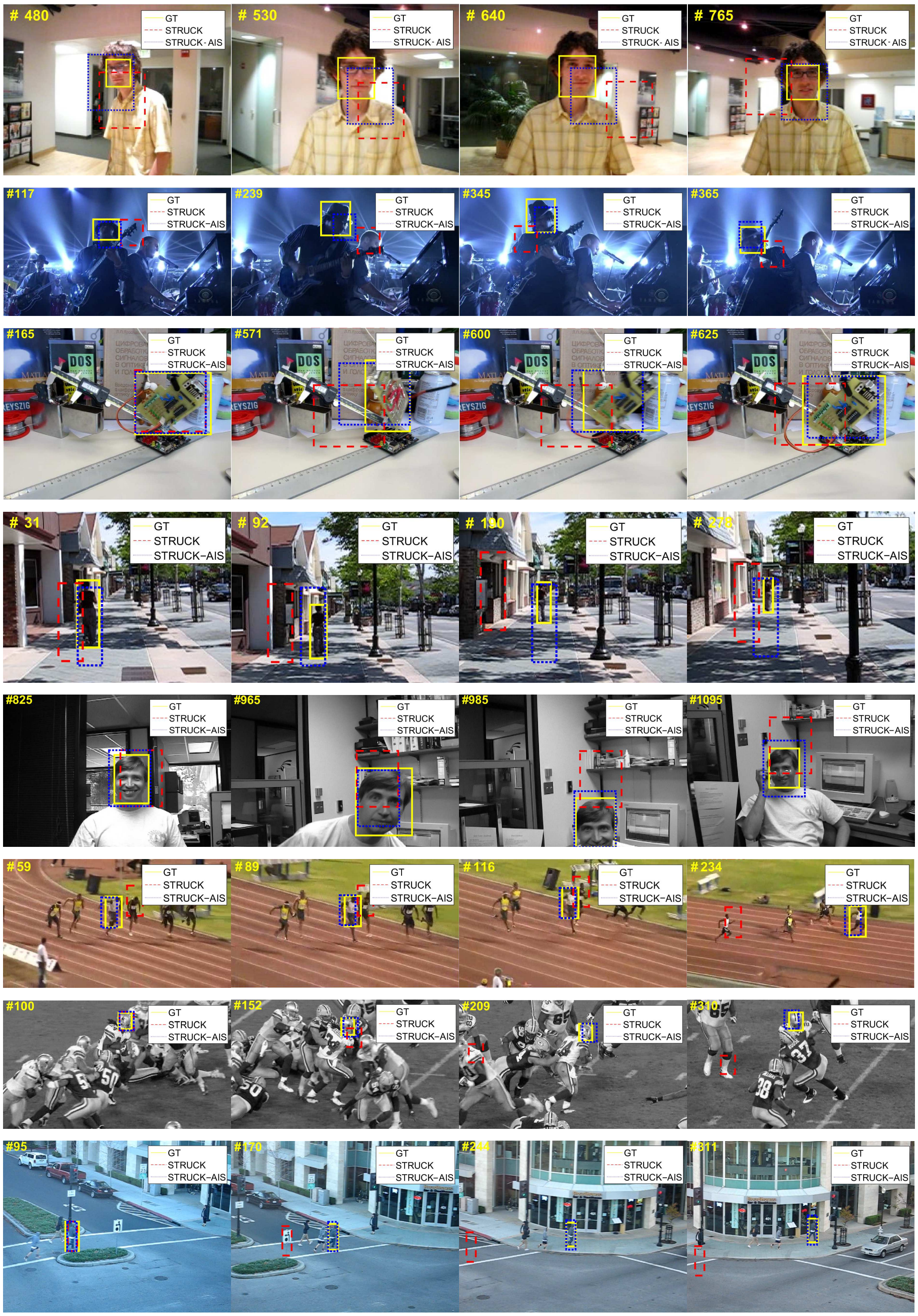


Figure 4.10: Subjective results of the proposed AIS method applied to STRUCK.

Table 4.1: Tracking quality of base and modified (AIS) STRUCK.

Sequence	AOR		CLE		FR	
	STRUCK	STRUCK-AIS	STRUCK	STRUCK-AIS	STRUCK	STRUCK-AIS
Deer	0.74	**0.75**	5.29	**4.89**	0.00	0.00
Walking2	0.51	*0.51*	12.13	**11.95**	0.00	0.00
Board	0.64	**0.69**	35.77	**24.31**	0.02	**0.00**
David	0.30	**0.42**	45.71	**26.50**	0.30	**0.11**
Singer1	0.36	*0.36*	12.97	**11.89**	0.00	0.00
Car4	0.49	*0.49*	8.12	8.16	0.00	0.00
Girl	**0.74**	0.72	**2.66**	3.20	0.00	0.00
Shaking	0.35	**0.41**	38.36	**25.18**	0.07	**0.00**
FaceOcc2	0.77	**0.78**	6.47	**6.10**	0.00	0.00
Boy	0.77	**0.78**	3.73	**3.43**	0.00	0.00
Sylvester	0.73	**0.73**	6.07	**5.95**	0.00	0.00
Dancer2	0.75	**0.76**	9.25	**8.80**	0.00	0.00
BlueCar1	0.80	**0.81**	4.88	**4.24**	0.00	0.00
Biker	0.25	**0.26**	25.55	**24.37**	0.50	**0.49**
Trellis	**0.61**	0.55	**6.13**	13.52	**0.01**	0.03
Car2	0.69	**0.69**	3.13	**2.83**	0.00	0.00
Human9	0.11	**0.18**	41.10	**32.03**	0.53	**0.27**
Dudek	0.65	**0.72**	29.19	**12.81**	0.05	**0.00**
Basketball	0.06	**0.32**	184.62	**79.20**	0.86	**0.500**
Diving	0.29	**0.33**	34.76	**24.34**	0.08	**0.00**
Bolt2	0.14	**0.42**	110.80	**14.23**	0.54	**0.00**
Human4	0.21	**0.40**	234.58	**79.90**	0.69	**0.37**
Football	0.32	**0.64**	139.42	**7.48**	0.57	**0.00**
Couple	0.51	**0.55**	22.92	**10.76**	0.17	**0.03**
Human6	0.2	**0.22**	102.57	**97.64**	0.56	**0.44**
BlurFace	0.593	**0.80**	29.80	**8.62**	0.00	0.00
Tiger2	0.367	**0.55**	46.18	**19.41**	0.17	**0.00**
BlurOwl	0.775	**0.81**	7.04	**5.56**	0.00	0.00
Crowds	0.088	**0.74**	374.76	**3.78**	0.87	**0.00**
BlurCar2	0.753	**0.763**	7.99	**6.31**	0	0
DragonBaby	0.233	**0.338**	64.18	**48.16**	0.314	**0.283**
Soccer	0.149	**0.167**	85.31	**81.02**	0.608	**0.571**
Freeman4	0.119	**0.224**	51.58	**17.69**	0.727	0.328
Ironman	0.036	**0..87**	171.84	**177.28**	0.858	**0.813**
Skating1	0.393	**0.49**	55.90	**7.85**	0.22	**0.00**
Average	0.44	**0.528**	57.73	**26.26**	0.248	**0.12**
Improvement	20%		54.5%		51.6%	

4.7 Conclusion

In this chapter, we use AIS-based optimization to adaptively select the best performing parameters of SVM of the STRUCK tracking model. The proposed method overcomes the bias of human intervention and the tedious work adopted for parameter selection. For tracking, the proposed AIS takes the advantage of the already generated sample patches in which their corresponding features are used for training. The obtained results show that the proposed framework applied to STRUCK tracker outperforms the original algorithm according to various objective measures. However, it works at low frame rate since AIS parameter optimization is done on-line. For future work, we plan to study the effect of using different features such as SURF, ORB, and HOG for the training of the complementary SVM model on the tracking accuracy. We plan to enhance the AIS model to achieve faster convergence rate in order to improve the speed of the proposed method.

Chapter 5

Artificial Immune System Based Parameter Optimization of Graph Cut Segmentation

5.1 Abstract

Little research attention has been given to study the use of optimization using artificial immune system in computer vision. Long and tedious trial and error approaches are usually used for the selection of object segmentation such as Graph Cut parameters for a specific application. However, these approaches do not guarantee best performing parameters. This chapter proposes the use of artificial immune system based optimization for adaptive selection of the near-optimal parameters of Graph Cut segmentation method. The obtained results show an enhanced segmentation quality, from points of view of precision and accuracy, and an enhanced object tracking quality in methods that use object segmentation for tracking.

5.2 List of Symbols

.

Symbol	Description
P_s	AIS population size
A_s	AIS Archive size
K_{max}	AIS maximum number of iterations
λ_A	Weighting term to control over and under segmentation
δ_A	Camera noise in G-Cut model
λ_r	Range of parameter λ
δ_r	Range of parameter δ
$\{\lambda_A, \delta_A\}$	G-Cut parameter set calculated by AIS model
E_G	Energy function of image I
I_g	Gound truth segmentation of image I
I_p	Intensity of pixel p
I_q	Intensity of pixel q
$B_{p,q}$	Boundary smoothness term of the (p,q) neighboring pixels
R_p	Regional term to encode the energy assigned to pixel p
t_p	Number of positive cases correctly classified
f_p	Number of positive cases incorrectly classified
f_n	Number of negative cases incorrectly classified
S_f	Interactive foreground seeds
S_b	Interactive background seeds
P	Precision measure of segmentation quality
A	Accuracy measure of segmentation quality

5.3 Introduction

Object segmentation has many uses in computer vision [117, 118]. Interactive segmentation has appealing results and is used in diverse applications [75–78]. There are different approaches to interactive segmentation, such as snakes [119], level sets [120], and Graph Cut (G-Cut) [75, 78]. A common problem in segmentation algorithms is the parameter selection that significantly affects the accuracy of results. Optimization techniques for the selection of near-optimal parameters is advantageous. First, it reduces the tedious experimental work spent for selecting the best performing parameters. Second, it has the ability to find the near-optimal parameters that lead to better segmentation quality.

G-Cut segmentation [75] has appealing results as it compromises between the computational complexity and the ability to achieve global solution for binary labeling [78]. Inappropriate choice of G-Cut parameters may result in unsatisfactory segmentation. This chapter proposes to use artificial immune system (AIS) based parameter selection of G-Cut segmentation algorithm off-line. AIS is used to find the parameters that simultaneously minimize both false positives and negatives based on ground truth, that in turn leads to better segmentation quality.

The rest of the chapter is organized as follows. Section 5.4 presents prior work and its relation to the proposed approach. Section 5.5 introduce the proposed approach. The experimental results are presented in Section 5.6, followed by a conclusion in Section 5.7.

5.4 Prior Work

Several approaches have addressed the selection of segmentation parameters in literature [62, 63, 121, 122]. Some of these approaches [62, 121, 122] cannot be applied to general scene images. In [121], Peng and Veksler presented a parameter selection approach for G-Cut segmentation using a measure based on different features that are combined using AdaBoost. The G-Cut is run for different parameter values and those of highest quality according to the learnt measure are selected. The efficiency of this approach depends on the number of images used in the training model and the accuracy of the manual labeling used. In [62], the authors investigated the use of AIS in aerial and satellite image segmentation. Although AIS can locate road pixel candidates using simple local detectors and a small amount of hand-classified sample imagery, such application-specific work has assumed high resolution images. Also, [62] did not verify the quality of the segmentation objectively. In [122], an adaptive parameter selection for cell segmentation using G-Cut is presented. The cell boundary extraction is fused into the G-Cut parameter value, which alter pixel weights in the graph formulation. This approach is designed for a specific type of images. For generic scene images, AIS is investigated in [63] for the selection of segmentation parameters where Cuevas et al. presented a multi-threshold mixture-of-Gaussian segmentation based on AIS. In this approach, 1D histogram of one image is approximated through a Gaussian mixture model whose parameters are calculated through AIS.

AIS is not yet, up to our knowledge, applied for optimization of G-Cut segmentation, of generic scene images, as we propose.

5.5 Proposed AIS-based G-Cut Optimization

5.5.1 Background

G-Cut represents an image I as a graph $G = (V, E)$, where V includes a set of nodes V_p corresponding to the pixels in I and another set of terminal nodes V_t that represent possible classes for labeling of image pixels, $V = V_p \cup V_t$. Edges E comprise a set of edges connecting adjacent pixels E_n, and a set of edges E_t connecting every pixel to terminal node that represents the corresponding pixel's affinity to each possible class label, $E = E_n \cup E_t$. The weights on these edges correspond to energies defined by a markov random field (MRF) formulation. G is partitioned into two separate groups called (cuts) C as a subset of edges that, when removed, separates the terminal nodes from each other. The cost of C is defined to be $|C| = \sum_{c \in C} W_c$, where W_c is the weight of the edge c. In G-Cut, the main interest is to find the cut of minimum cost by applying the max-flow min-cut algorithm [123]. Given an image I with set of pixels $\{p\}$, the goal is to assign the most likely label $l \in L = \{l_{fg}, l_{bg}\}$ for each pixel. It is assumed that I has labeling that is consistently smooth along the image with discontinuity at boundaries. The smoothness as well as the prior knowledge (strokes) that is automatically generated are encoded as a function of energy E_G over I, where E_G is a linear combination of boundary smoothness term $B_{p,q}$ over each pair of neighboring pixels p, q and regional term R_p that encodes the energy assigned to pixel p of label l. The solution to the segmentation problem is assumed to be the joint assignment of pixels $\{p\}$ to labels $l \in L$ in a way that minimizes the energy function E_G formulated as

$$E_G = \sum_{(p,q) \in N} \lambda_A B_{p,q} + \sum_{p \in P} R_p, \tag{5.1}$$

where λ_A is a relative weighting term and $B_{p,q}$ is the boundary smoothness term that assigns high energy to adjacent pixels of similar intensity to encourage them to belong to the same side of cut C and the energy decreases as the pixels become more dis-similar according to

$$B_{p,q} = exp\left(\frac{-(I_p - I_q)^2}{2\delta_A^2}\right), \tag{5.2}$$

where I_p and I_q are the intensities of p and q pixels, respectively, and δ_A represents camera noise. Such formulation penalizes the separation between pixels of similar intensity ($|I_p - I_q| < \delta_A$).

5.5.2 Our Approach

We address the use of clonal selection-based AIS optimization [59] to find the best parameters for G-Cut segmentation off-line. As an objective function, we used both precision and recall measures [124]. Given the ground truth and the output segmentation, we construct a confusion matrix that represents the number of positive t_p, negative t_n cases correctly classified, and the number of negative f_n, and positive f_p cases incorrectly classified. We use this matrix under the supervision of AIS to estimate the best segmentation parameters as illustrated next.

We estimate both λ_A and δ_A parameters in (5.1) and (5.2) using the clonal selection-based AIS optimization [59] by maximizing the precision and recall measures simultaneously in order to achieve the best segmentation quality according to

$$\begin{aligned} (\lambda_A, \delta_A) = \underset{(\lambda,\delta)}{\text{maximize}} \quad & \left(\frac{t_p}{t_p + f_p} . \frac{t_p}{t_p + f_n}\right) \\ \text{s.t.} \quad & \delta \in \delta_r \quad \text{and} \quad \lambda \in \lambda_r, \end{aligned} \tag{5.3}$$

where t_p, f_p, and f_n are the numbers of true positives, false positives, and false negatives, respectively. $\lambda_r = [1, 255]$ and $\delta_r = [0, 10]$ are the search ranges used by AIS optimization for both λ and δ parameters. The goal of the AIS optimization is to find the (λ, δ) pair that maximizes both the precision and recall measures simultaneously to achieve the best segmentation quality. Algorithm 5.1 shows the detailed steps of our AIS optimization process. The detailed flowchart of the optimization process is shown in Figure 5.1. Note that in our simulation, the seeds S_f and S_b are selected interactively at first time and stored in off-line file and then re-loaded for any use of segmentation for fair results.

AIS algorithm starts by randomly generating an initial population with number of P_S antibodies (ab) as pairs of (λ, δ) according to the selected ranges δ_r and λ_r of δ and λ, respectively. Such population is considered as an initial generation to update the history of solution during the optimization process.

Algorithm 5.1: AIS-based optimization of G-Cut parameters.

Data: Population size P_S; Archive size A_S; Maximum iterations K_{max}; Parameters ranges λ_r and δ_r; image I; image ground truth segmentation I_g

Result: Best parameter pair $(\lambda_A \ , \ \delta_A)$

Select interactive seeds S_f and S_b ;
History = initial population $\{(\lambda_i \ , \ \delta_i)\}$; i=1, ..., P_S;
for k = *1 to* K_{max} **do**
    $Output$ = G-Cut$(I, S_f, S_b, (\lambda_i \ , \ \delta_i))$; $\forall i$;
    $\{a_i\}$ = Evaluate($Output, I_{gi}$); $\forall i$;
    Dominance ($\{a_i\}$);
    $\{(\lambda_i \ , \ \delta_i)\}_{clone}$ = Clone ($\{(\lambda_i \ , \ \delta_i)\}$) $\forall i$;
    Archive = Update-Archive ($\{(\lambda_i \ , \ \delta_i)\}_{clone}$) ;
    $\{(\lambda_i \ , \ \delta_i)\}_{binary}$ = Encoding ($\{(\lambda_i \ , \ \delta_i)\}_{clone}$) $\forall i$;
    $\{(\lambda_i \ , \ \delta_i)\}_{mutate}$ = Mutation ($\{(\lambda_i \ , \ \delta_i)\}_{binary}$);
    History = Update-History ($\{(\lambda_i \ , \ \delta_i)\}_{mutate}$);
    $k = k + 1$;
$(\lambda_A \ , \ \delta_A)$ = Max (Archive);

We calculate the value of the objective function (5.3) by applying the segmentation using the candidate (λ, δ) pairs. The affinity $\{a_i\}$ between the antibodies and antigens is obtained by calculating the product of both precision and recall to maximize the segmentation quality by minimizing f_p and f_n simultaneously. The dominance of ab is checked to obtain the best (λ, δ) pair that best matches the objective function as in (5.3). The cloning process clones the non-dominated solution as well as the dominated ones in order to ensure the diversity in the optimization. To this end, the non-dominated ab (that perfectly match the objective function) are selected as the best ab to update the archive of best solutions. Then, each ab is encoded as a binary string and the mutation process is applied. At the end of each generation, the archive is updated with best solutions and the history of solutions is updated. After K_{max} iterations, the algorithm terminates and the best parameter pair $(\lambda_A \ , \ \delta_A)$ is found as the pair of highest objective function in the archive.

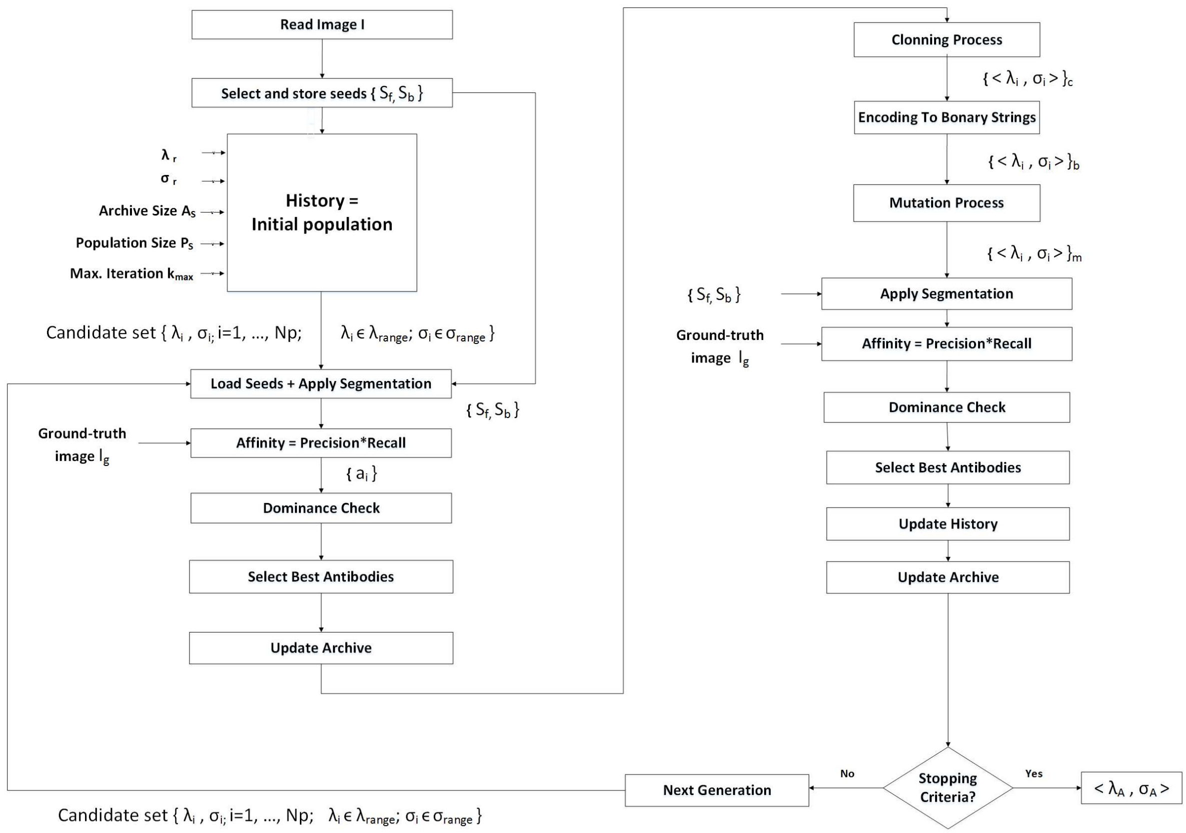


Figure 5.1: Block diagram of the proposed AIS-based G-Cut optimization.

5.6 Results

5.6.1 Experimental Setup

We developed our proposed AIS model in section 4.6.1 in a way so it can be used for parameter optimization of any computer vision/video processing method with its own objective function. Accordingly, for experiments in this chapter, the AIS model parameters are the same as in chapter 4: Population size $P_S = 20$; Archive size $A_S = 4$; Maximum iterations $K_{max} = 20$.

The performance of the object segmentation, with default and AIS parameters, is measured using the precision P and accuracy A metrics [124], defined as

$$P = \frac{t_p}{t_p + f_p} \quad \text{and} \quad A = \frac{t_p}{t_p + f_p} \cdot \frac{t_p}{t_p + f_n}. \tag{5.4}$$

t_p is the number of positive cases correctly classified, f_n is the number of negative cases incorrectly classified, and f_p is the number of positive cases incorrectly classified. High values of P and A indicate good segmentation. The preformance of object tracking is measured using well-known AOR, FR, and CLE measures.

5.6.2 Objective and Subjective Results

To verify the effectiveness of the proposed AIS approach, we first use precision and accuracy metrics to measure the improvement in segmentation quality using the proposed AIS approach. Then, we integrated the G-Cut into a segmentation-based tracking method (such as in chapter 3) and verify the tracking quality with and without AIS selected parameters, using the tracking performance metrics AOR and FR (defined in section 2.3).

Figure 5.2 shows an example plot for the iterative AIS process to select the best performing parameter set, that has the highest recall and precision measures, for the image shown in Figure 5.3 (left). AIS generates candidate parameter sets and evaluates the corresponding recall and precision measures (red circles). Several parameter sets can lead to similar segmentation quality. The AIS search strategy proceeds to find parameter sets of higher precision and recall measures, and selects the best performing parameter set (top right corner black filled circle) that maximizes both precision and recall measures simultaneously.

Figure 5.3 (middle) shows the segmentation result using the achieved parameter set compared with the ground-truth image shown in Figure 5.3 (right).

As shown in Figures 5.4 and 5.5, the segmentation results using the obtained AIS-parameters, for 50 images from DUT-OMRON data set [125], show better quality according to the precision and recall measures (the higher the better) than using the default parameters. The proposed method achieved 14.31% improvement according to the P measure, and the A measure shows an improvement of 8.65%.

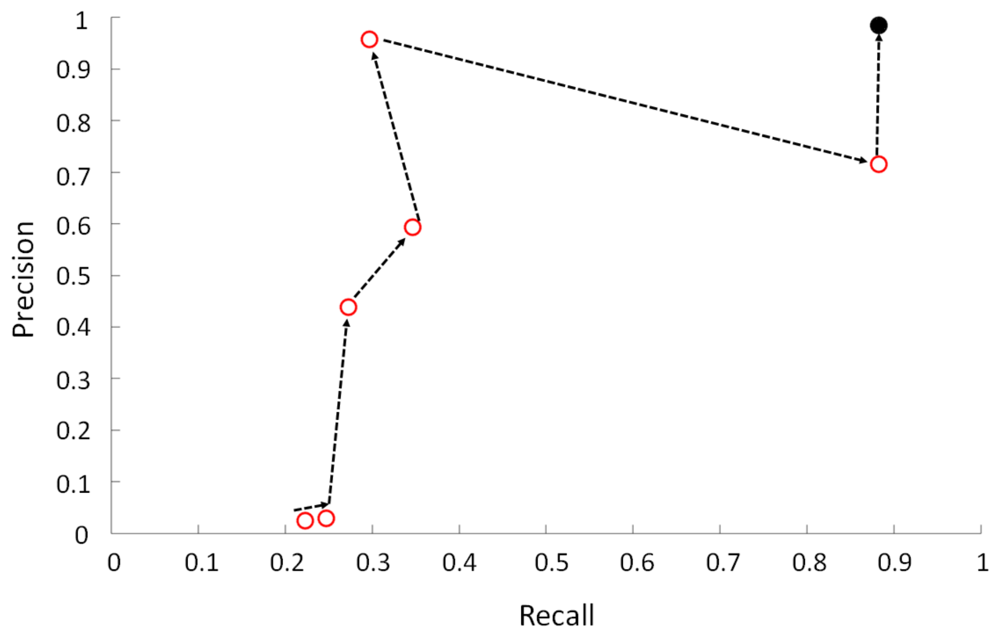


Figure 5.2: AIS selection of the best performing segmentation parameters.

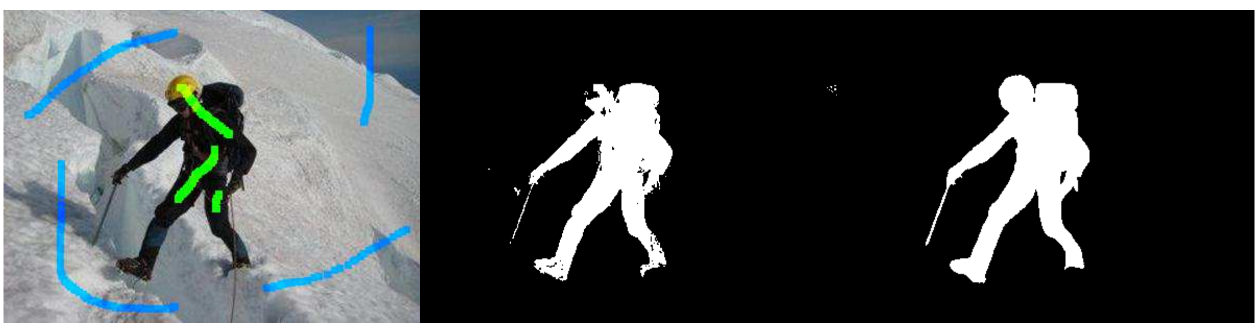


Figure 5.3: Example of G-Cut-AIS-based segmentation parameters: input image and seeds (left); segmentation using AIS selected parameters (middle); ground truth (right).

Figure 5.6 gives examples of the enhanced segmentation quality gained by the proposed AIS-based G-Cut parameter optimization, from the DUT-OMRON data set [125]. Figure 5.6 (1^{st} column) shows four original images with interactive selected seeds, (2^{nd} column); segmentation results using the default parameters, (3^{rd} column); segmentation results using AIS-parameters, and (4^{th} column); the corresponding ground truth images.

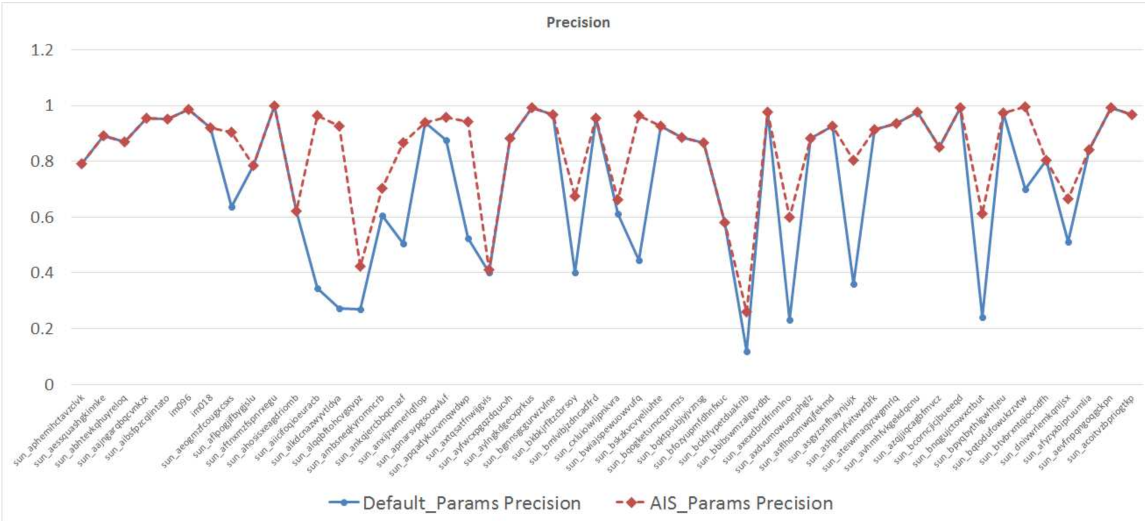


Figure 5.4: Precision plot of the default and AIS-based parameters of G-Cut segmentation.

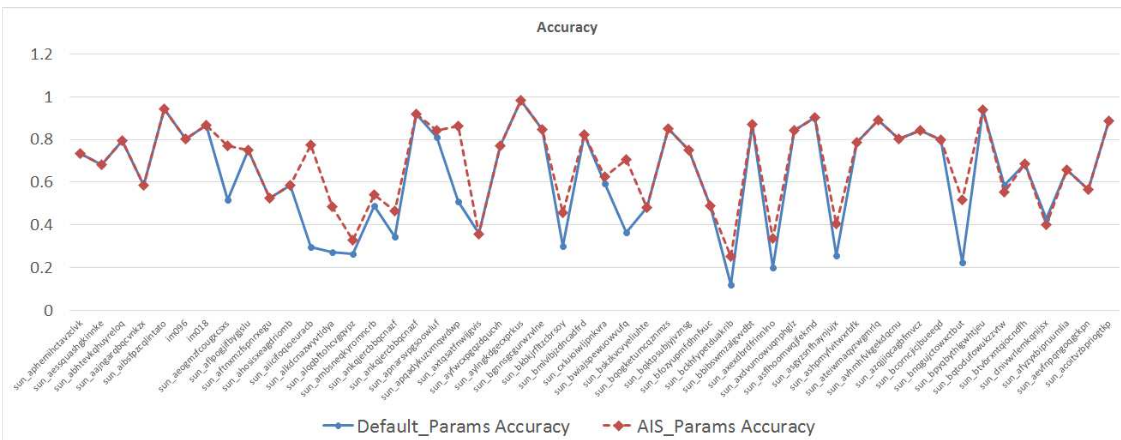
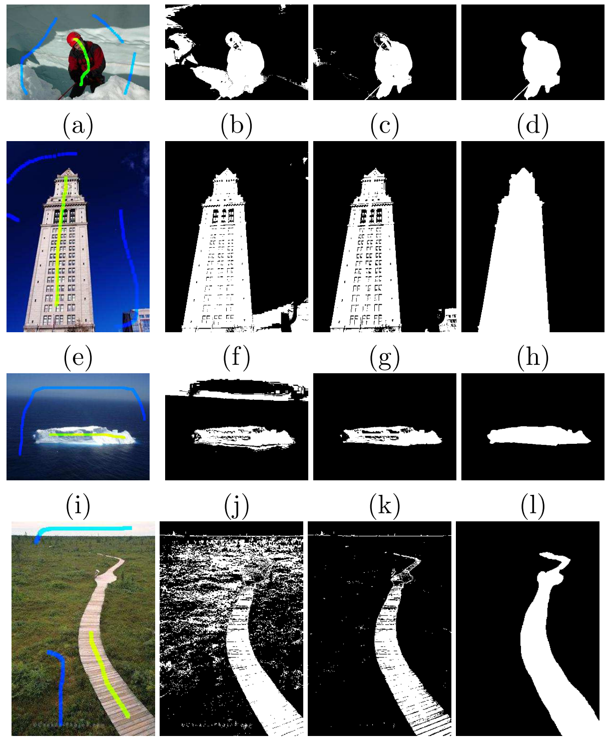


Figure 5.5: Accuracy plot of the default and AIS-based parameters of G-Cut segmentation.

We integrated G-Cut with the AIS selected parameters into our DDC approach in Chapter 3 in order to evaluate the achieved tracking improvement compared with the default G-Cut parameters. The improvement in AOR, CLE, and FR over all 100 test sequences of OTB benchmark [56] in Figures 5.7 and 5.8 show that using AIS-based segmentation parameters lead to better tracking quality for both STRUCK [2] and STAPLE [3] trackers. The same tracking environment is used for both tests and the only change is the segmentation parameters.



(a) (b) (c) (d)
(e) (f) (g) (h)
(i) (j) (k) (l)
(m) Seeds (n) Default (o) AIS (p) Ground-truth

Figure 5.6: G-Cut segmentation samples using default vs. AIS parameters.

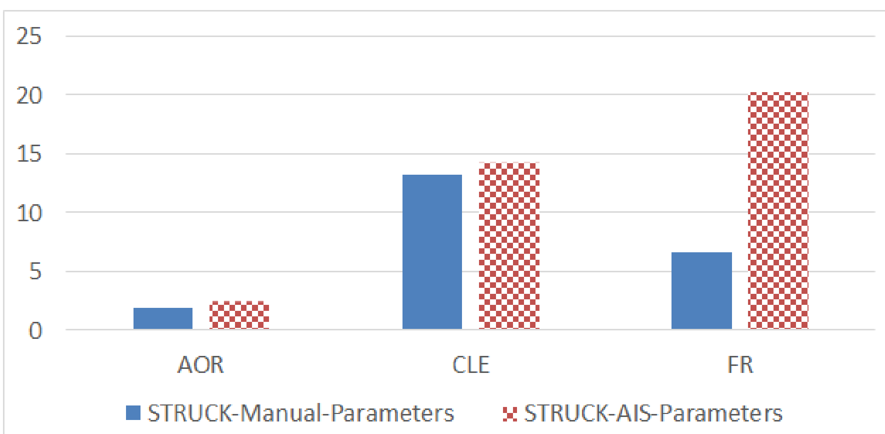


Figure 5.7: Tracking improvement (in %) of AIS-based vs. manual segmentation parameters when using STRUCK [2]

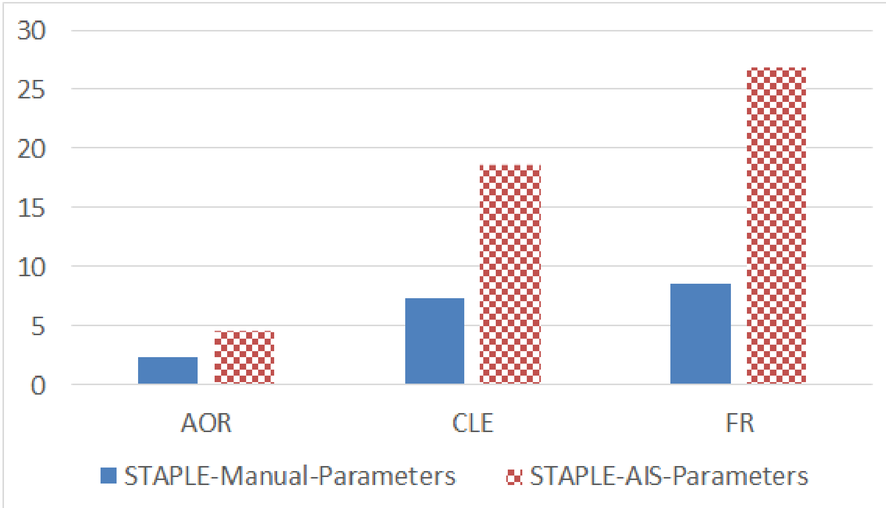


Figure 5.8: Tracking improvement (in %) of AIS-based vs. manual segmentation parameters when using STAPLE [3]

5.7 Conclusion

In this chapter, artificial immune system based optimization is used to adaptively select the best performing parameters of Graph Cut segmentation offline. The proposed method overcomes the bias of human intervention and the tedious work for parameter selection. A better segmentation quality and a better tracking quality have been verified according to related objective measures. The obtained results show that the proposed artificial immune system can effectively serve as an attractive tool for the selection of the best performing Graph Cut segmentation parameters.

Chapter 6

Robust Scoring and Ranking of Object Tracking Techniques

6.1 Abstract

Object tracking is an active research area and a large number of tracking techniques, or trackers for short, have been proposed recently with demonstrated success. When introducing a new tracker, its quality is compared against existing trackers based on objective performance measures. Recent studies have shown that these performance measures are correlated and cannot reflect the different aspects of tracking performance when used individually. In addition, they do not use robust statistics to account for the presence of outliers that might lead to insignificant results. This chapter presents a framework for scoring and ranking of trackers using known quality metrics (overlap ratio and failure rate), coupled with a robust estimator against dispersion and outliers (median absolute deviation). We also propose a unified (overlap and failure) performance index, as well as recovery, and pure drift measures to facilitate the evaluation process. Ten state-of-the-art different performing trackers are scored and ranked using the proposed framework on a public benchmark of 100 video sequences. The obtained results show how the proposed framework facilitates the evaluation of the relative performance of different trackers.

6.2 List of Symbols

Symbol	Description
IOF	Unified performance index
N_l	Number of frames for sequence v_l
n_r	Number of frames that tracker successfully recovered from failure
R	Recovery measure of tracker t_i on test sequence v_l
D	Drift measure of tracker t_i on test sequence v_l
μ_{ij}	Mean quality for tracker i as an average over all sequences according to metric j
N	Number of trackers
N_v	Number of test sequences
N_p	Number of metrics
$\{r_{ij}\}$	Rank vector for all ranked trackers
$\{s_{ij}\}$	Score vector for all scored trackers
d_q	Median absolute deviation threshold

6.3 Introduction

Video object tracking plays an important role in an increasing number of applications like augmented reality, visual surveillance, and robotics. A variety of trackers (such as STRUCK [2], ASLA [8], SCM [9], KCF [10], SAMF [11], LOT [12], DSST [13], Staple [3], TCNN [83], and CCOT [84]) have been developed to investigate challenges related to object tracking with their code available for evaluation. Evaluation and ranking of tracking algorithms (trackers) are of critical importance for both the comparison and further development and enhancement of algorithms. A variety of benchmark papers have been presented to evaluate the performance of trackers [16, 56, 57, 91–94].

One of the important features of a good performance model is the robustness against both outliers and deviations from model assumptions. In statistics, outliers are observation points that are distant and do not fit other observations. They usually result from variability or randomness in measurements or model assumptions that occur commonly in different tracking algorithms, or it may indicate experimental error [126]. Classical estimation methods rely heavily on assumptions such as the normal distribution of errors as in statistical mean. Unfortunately, when there are outliers in the data, such estimators often have very poor performance, when judged using known measures such as the breakdown point (BP), which is one of the most popular measures of both reliability and robustness of a statistical procedure [127, 128]. It is often the first and most important number to be looked at before going into the details of local robustness properties. Among scale estimators, range, standard and mean deviation all have $BP = 0$, the interquartile range has $BP = 1/4$, while the median has $BP = 1/2$. The counterpart of the median among scale estimators is the median absolute deviation (MAD) [129]. MAD, the median of the absolute differences of the data elements from their median, represents a useful basis for reliable rejection of outliers [130]. This chapter proposes a strategy for effective scoring and ranking of different trackers that uses the MAD to effectively quantify the statistical dispersion in a given set of numerical (quality) data.

Several performance metrics of the tracking algorithms use empirical discrepancy methods [54] that compare off-line ground-truth data with the estimated trajectories. Among such metrics, the average overlap ratio (accuracy), center location error, failure rate (robustness), or derivatives thereof, such as success and precision plots are commonly used [55]. Recent studies [55, 131] have shown that different measures are correlated and cannot reflect different aspects of tracking performance when used individually. Investigating the correlation between different performance measures, Cehovin et al. in [131] concluded that the average overlap ratio on re-initialized trajectories", and the failure rate are the least correlated. The basic performance measures used in recent evaluation benchmarks [56, 91, 92] are accuracy, that measures how well the predicted bounding box (BB) overlaps with the ground truth, and robustness, that measures how many times the tracker loses the target (fails) during tracking based on the overlap ratio. This chapter proposes new performance measures (drift, recovery, and unified index) to evaluate and rank trackers.

The rest of the chapter is organized as follows. In Section 6.4, we present related work and differentiation to our approach. Section 6.5 introduces the proposed ranking measures. Section 6.6 presents the proposed evaluation framework. Section 6.7 discusses the obtained results. Section 6.8 concludes the work.

6.4 Prior Work

6.4.1 Ranking Methods

Prior work can be divided into ”evaluation” and ”ranking” methods. Evaluation methods [56, 57, 94] report values of performance measures; ranking methods [91–93, 96, 132] rank trackers using some scale. Wu et al., in [56], used the precision and success plots for overall performance evaluation of trackers in addition to temporal robustness evaluation (TRE). As trackers are sensitive to initialization, TRE starts the tracking algorithms at different frames (temporally) and evaluates the tracking performance accordingly. In [57], the authors proposed the one-pass evaluation with restart (OPER) measure that re-initializes the tracker once it fails during tracking. In addition, they proposed the spatial robustness evaluation with restart (SRER) that evaluates how the tracker is sensitive to spatial perturbations during tracking. In [94], Li et al., evaluated trackers via online system based on criteria to calculate the overlap ratio based on occlusion level that is incorporated as a criterion for performance evaluation with the NUSPRO database. In addition, as a criterion for computing the overlap, they used different thresholds to determine whether a frame is successfully tracked.

In VOT2013 [91], 27 trackers were evaluated and ranked by averaging the performance on test sequences using accuracy and robustness measures. A tracker was re-initialized several frames after a failure occurs. They evaluated each tracker separately for each performance measure on each attribute sequence, and by averaging the evaluations over the different attributes, the ranking with respect to such performance measure was obtained. In [132], Pang and Ling used a ranking approach to analyze the reported results of different trackers. A shortcoming of such approach is the limited experimental evaluations in terms of number of sequences and performance metrics used. In VOT2014 [92], M. Kristan et al. used the same methodology for ranking in [91] and introduced a new unit for reporting the tracking speed without being influenced by the used hardware. VOT2015 [93] combined the raw values of per-frame accuracies and failures for evaluation. The VOT2016 [96] used accuracy, robustness, and speed for ranking. VOT2016 also introduced a sub-challenge (VOTTIR2016) to tracking in thermal and infrared imagery.

In some applications, outliers can be subjectively removed from the sampled observations [133]. However, such subjective selection of outliers is difficult. In [91], authors mentioned that a significant novelty of the proposed evaluation protocol is that it explicitly addresses the statistical significance of the results. However authors do not tackle the presence of outliers that may lead to insignificant results. In [91–93], the authors run each tracker on each sequence 15 times to obtain a better statistic, using the mean to average such multiple results. However, the mean is not robust against outliers. To our knowledge, the state-of-the-art ranking methods [91–93, 96] do not incorporate a robust statistical measure to account for the presence of outliers that are commonly to accrue in most of the tracking algorithms.

To summarize, recent ranking methods have three limitations; First, they use evaluation measures that can be correlated to some extent and hence, will not reflect the different aspects of tracking performance. Second, such measures can be highly affected with outliers in the data set. Third, they do not use a robust statistical measure to account for the presence of outliers that might lead to insignificant results, that in turn will affect the ranking accuracy.

This chapter presents a framework for scoring and ranking of different trackers using less correlated quality metrics (overlap ratio and failure rate), coupled with a robust statistical estimator (MAD) against dispersion and outliers.

6.4.2 Related Performance Measures

Numerous performance measures have been proposed in literature for the evaluation of object tracking algorithms. Such measures range from basic ones like center location error [134], overlap ratio, tracking length, failure rate, and Fscore [55], to more sophisticated measures, such as CoTPS [135], which combines several metrics. An advantage of the combined measures is that they provide a single score to rank different trackers. However, they are not always easily interpretable.

Nawaz and Cavallaro [135] proposed a hybrid measure called the Combined Tracking Performance Score (CoTPS) that combines the information on tracking accuracy and failure into a single score as a weighted sum of accuracy and failure scores. However, as per [55], the proposed CoTPS combines the quality measures in a rather complicated manner, prohibiting a straightforward interpretation.

Different from related work, we propose a performance index, that combines accuracy (overlap ratio) and robustness (failure rate) in a unified measure, as a novel means for ranking of object trackers. In addition, we propose two new performance measures; recovery and drift, that rank of different trackers in these two aspects.

6.5 Proposed Performance Measures

In this section, we introduce a unified performance index that combines both overlap ratio and failure rate into one metric to facilitate ranking of different tracking algorithms. We also present two new metrics; recovery and pure drift metrics; as two separate ranking measures that support the proposed index measure.

6.5.1 Proposed Unified Performance Index

An ideal tracker creates a BB which is completely coinciding with that of the ground truth ($AOR = 1$). In addition, it continuously tracks the object without failure ($FR = 0$). Our performance index IOF is defined as

$$IOF = AOR \cdot (1 - FR). \tag{6.1}$$

Ideally $IOF = 1$, and the worst case has $IOF = 0$. Figure 6.1 illustrates the behavior of both the proposed IOF index and CoTPS measure index [135]. Different than the complicated CoTPS measure, as shown in [55], the proposed index can be easily interpreted. Figure 6.1 shows that IOF reaches its highest quality when the AOR is maximized and FR is minimized, which confirms both measures theoretically. However, CoTPS can achieve its highest quality when AOR is maximized regardless of FR.

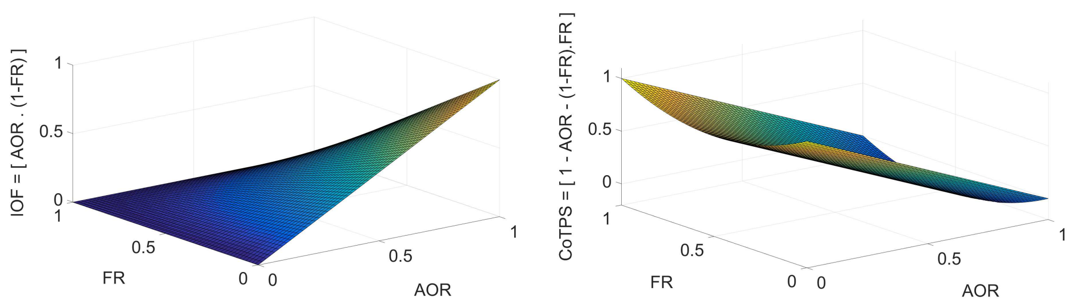


Figure 6.1: The behavior of the proposed IOF performance index (left) versus CoTPS measure (right).

6.5.2 Proposed Recovery and Drift Measures

For a tracking algorithm, the ability to recover after the occurrence of a failure (or drift) is important, particularly when the tracking algorithm is evaluated among set of other ones. We define recovery R of tracker t_i on test sequence v_l, as the number of frames the tracker successfully recovers from failure, meaning the tracker went from $AOR = 0$ to $AOR \neq 0$ over the whole video sequences. We define drift R as the number of frames a tracker goes from the state $AOR \neq 0$ to $AOR = 0$, over the whole video sequence. We define pure drift (or pure recovery) (pRD) per a video sequence as

$$pRD = \frac{R - D}{N_l - 1}. \tag{6.2}$$

where N_l is the total number of frames in video sequence v_l. Algorithm 6.1 illustrates calculating recovery R, drift D, and pure drift pRD measures. At frame F_t, using ground truth data AOR is 0 if no overlap occurs between the output BB B_t and ground truth BB B_t^g. We start the algorithm from frame F_2 as the first frame is the ground-truth data (no drift). Then the recovery and drift events are checked using the AOR data between two consecutive frames F_t and F_{t-1} and the corresponding counters (D) and (R) are updated. The pure drift (pRD) is calculated as in (6.2) and is between 1 and -1, where negative values mean the tracker went through more drifts than recoveries and positive values mean the tracker, well recovered from drifts over the whole video sequences. A good tracker should give high positive pRD values: $pRD = 1$ indicates the best performance since the tracker shows no drift throughout the video sequence;

$pRD > 0.333$ indicates good performance, meaning the tracker shows more recoveries than drifts.

Algorithm 6.1: Recovery and drift calculation of a tracker on sequence l.

Data: Quality data AOR for a tracker and sequence v_l with N_l frames
Result: Recovery R, drift D, and pure drift pRD measures

$D = 0$;
$R = 0$;
for *frames* $F_t = 2$ *to* N_l **do**
    if ($AOR_t \,!= 0 \wedge AOR_{t-1} \,!= 0$)
        $R++$;
    else if ($AOR_t == 0 \wedge AOR_{t-1} \,!= 0$)
        $D++$;
    else if ($AOR_t \,!= 0 \wedge AOR_{t-1} == 0$)
        $R++$;
    else if ($AOR_t == 0 \wedge AOR_{t-1} == 0$)
        $D++$;
    end
$pRD = (R - D)/(N_l - 1)$;

6.6 Proposed Evaluation Framework

6.6.1 Overview

The proposed framework comprises three steps as shown in Figure 6.2: data entry, scoring, and sequence-pooled ranking. The data entry step reads the following: a) output bounding box (BB), for all N trackers, $t_i, i = 1, \cdots N$; b) the performance evaluation metrics $p_j, j = 1, \cdots N_p$, N_p is the number of metrics used to calculate the quality values q_{ijl} for tracker t_i according to metric p_j over a test sequence $\{v_l; l = 1, \cdots, N_v\}$, N_v is the number of test sequences; c) the dispersion metric to use; and d) the ground truth BBs. For each p_j, for each v_l, the scoring step applies the dispersion measure on the vector q_{ijl}, for the N trackers and assigns score vector $\{s_{ij}, i = 1, \cdots N\}$ for each t_i according to p_j as the sum of scores over all sequences where $1 \le s_{ij} \le N_v$. Then, the ranking step calculates the mean values of q_{ijl} over all sequences based on p_j separately and applies the MAD dispersion measure to assign rank vector $\{r_{ij}, i = 1, \cdots N\}$ to all N trackers according to metric p_j, where $1 \le r_{ij} \le N$. We use MAD dispersion measure of the quality vector q_{ijl} of tracker t_i for sequence v_l using performance p_j as

$$d_q = MAD\{q_{ijl}\} = Median(\ |q_{ijl} - Median(\{q_{ijl}\})|\ ) \quad i = 1, \cdots N. \tag{6.3}$$

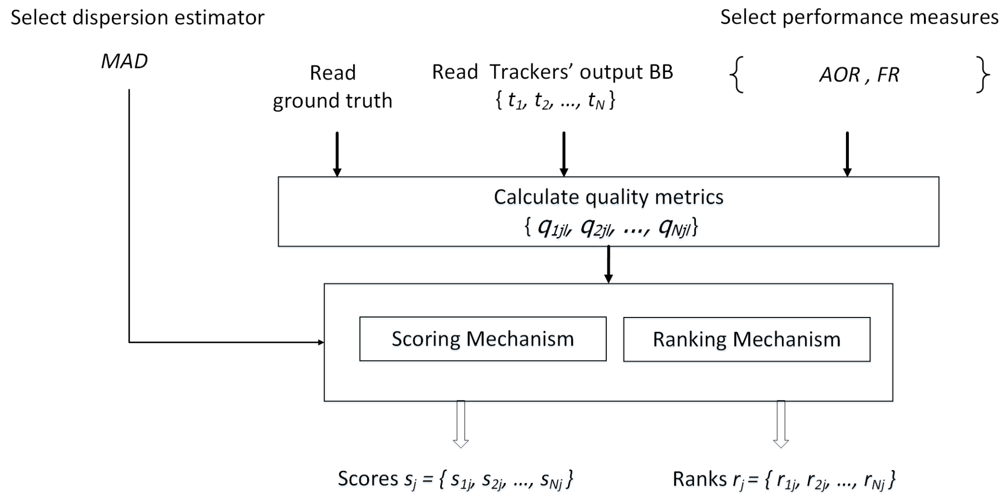


Figure 6.2: Block diagram of proposed scoring and ranking framework.

6.6.2 Scoring Mechanism

Due to outliers, counting the number of best and second best scores in numerical data of a tracker's measure is not reliable to measure performance, as doing so neglects the deviation that may exist in the data. For each test sequence, we, therefore, define a deviation threshold d_q based on the MAD dispersion as in (6.3), which evaluates a set's close affiliation to either a best score or a second best score. The process of the scoring mechanism for the N trackers over all $\{v_l\}$ sequences can be summarized in Algorithm 6.2. The video sequences are processed sequentially, and for every sequence, we score all trackers, and count all scores over all sequences for a specific quality metric. In Algorithm 6.2, $Best\{q_{ijl}; i = 1, \ldots, N\}$ is the *best* value (e.g., the maximum value for AOR), among the quality values of all N trackers for a v_l; $\{s_j\}$ is a vector of scores for the N trackers over all v_l for a p_j; and $Scores$ is the final scores for the N trackers according to metric p_j. For the quality values of the N trackers, scores are summed for each tracker over all sequences to find scores ($Scores$) for each performance measure p_j. Then, the tracker(s) with the maximum count is (are) selected as the best tracker(s).

The rest of the trackers that are not scored as best contributes to another round to choose the second best tracker(s) using same Algorithm 6.2 with a new threshold d_q selected based on the remaining trackers' quality values. This is repeated till all N trackers are scored.

Algorithm 6.2: Scoring of N Trackers for a metric and all test videos.

Data: Quality data q_{ijl}; $\{i = 1, \cdots, N\}$ of all test sequences $\{v_l; l = 1, \cdots, N_v\}$ for a metric p_j

Result: $Scores = \{s_{1j}, \cdots, s_{Nj}\}$ for a tracker t_i and a p_j

for *a performance measure j* **do**
    for *a tracker i* **do**
        $s_{ij} = 0$;
    for *a test sequence l* **do**
        $d_q = MAD\{q_{ijl}\}$;
        $O = Best\{q_{ijl}\}$;
        for *a tracker i* **do**
            if $|q_{ijl} - O| < d_q$
                score = 1;
            else
                score = 0;
            end;
            $s_{ij} = s_{ij}$ + score;
    $Scores = \{s_{ij}\}$;

6.6.3 Sequence-pooled Ranking Mechanism

All trackers are ranked according to their mean over all sequences. The ranking of N trackers is performed as follows. Given q_{ijl} of all v_l, the mean quality μ_{ij} for each tracker t_i is calculated as an average of q_{ijl} over all test sequences according to performance measure p_j. At the beginning, all trackers are marked as unranked and each tracker keeps contributing to the ranking process until it is assigned a rank. The best mean value O among all unranked trackers is selected according to p_j (e.g., the maximum value for AOR), and any tracker t_i that has a mean quality closer to the best O within d_q is marked as a ranked tracker and assigned a first rank level in the first round. The same process is repeated but only for the rest of the unranked trackers and a counter ($count_j$) is incremented at each round to keep the rank level updated. The ranking for the N trackers is summarized in Algorithm 6.3, where $Ranks_j$ is the rank vector assigned to all

N trackers according to p_j. In the first round, the quality of the N trackers is compared with the best quality of all trackers O_j according to metric p_j, and the closest ones, within a specific threshold d_q, are ranked the first. The counter $count_j$ is incremented by 1 and the rest of the trackers that are not ranked contributes to the next round to assign the second rank in a similar way but with a threshold d_q calculated based on the quality values q_{ijl} of the remaining unranked trackers. This is repeated until all N trackers are ranked.

Algorithm 6.3: Ranking of N Trackers.

Data: Quality data q_{ijl}; $\{i = 1, \cdots, N\}$ of all test sequences $\{v_l; l = 1, \cdots, N_v\}$ for a metric p_j
Result: $Ranks_j\{r_{1j}, \cdots, r_{Nj}\}$ for each tracker t_i and each p_j
for *a performance measure* j **do**
    for *a tracker* i **do**
        $r_{ij} = 0$;
    $count_j = 0$;
    $\{\mu_{ij}\}$ = average of $\{q_{ijl}\}$ $\forall i, l$;
    do
        $d_q = MAD(\{\mu_{ij}\})$ of unranked trackers i;
        $O = Best(\{\mu_{ij}\})$;
        for *each* μ_{ij} *of unranked tracker* i **do**
            if $|\mu_{ij} - O| < d_q$
                $r_{ij} = count_j + 1$;
                mark tracker i as ranked;
            end
        $count_j = count_j + 1$;
    while *There exist unranked trackers*;
    $Ranks_j = \{r_{ij}\}$;

6.7 Results and Analysis

6.7.1 Experimental Setup

We run each tracker with its published source code and default parameters, five times on each video sequence to obtain fair statistics. We have tested our framework with ten known trackers (STRUCK [2], ASLA [8], SCM [9], KCF [10], SAMF [11], LOT [12], DSST [13]), STAPLE [3], TCNN [83], and CCOT [84] using five performance measures (AOR, FR, IOF, R, and D), MAD dispersion estimator, and 100 video sequences [56] (that include the 11 challenges: illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutters, and low resolution). We tried two other dispersion measures, interquartile range and median maximal distance, and found that the scores they assigned to trackers do not well discriminate among trackers; but MAD does. In order to evaluate different trackers, we used the actual AOR and FR measures as a reference measure for the *true* order of all trackers. The plot of AOR vs. FR of each tracker, Figure 6.3 reflects the *true* tracking quality of all trackers, and will be used to support the validity of our new measures IOF, R, D. According to Figure 6.3, we can rank the ten trackers as follows: 1. CCOT, 2. TCNN, 3. STAPLE, 4. SAMF, 5. DSST, 6. KCF, 7. STRUCK, 8. SCM, 9. ASLA, 10. LOT.

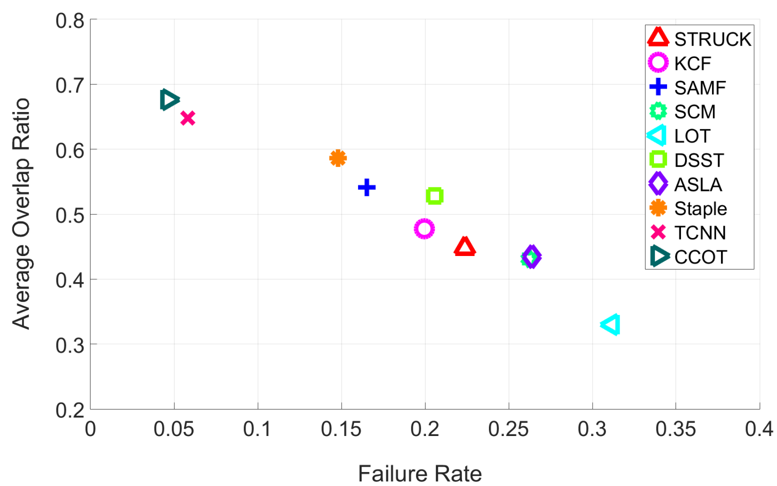


Figure 6.3: AOR vs FR as a reference measure to evaluate the proposed measures.

In addition to the AOR and FR individually as per table 6.1, our obtained IOF results showed that the ranking according to VOT2013 [91] contradicts with the reference measure of Figure 6.3, as it swapped the ranking of STRUCK (ranked 6^{th}) and KCF (ranked 7^{th}). On the contrary, the proposed IOF is found to match perfectly the ground-truth results in Figure 6.3.

6.7.2 Scoring and Ranking using AOR and FR

Table 6.1 shows the scoring and ranking results of the ten tested trackers based on AOR and FR individually. As shown, the number of best and second-best scores over all video sequences are assigned to each tracker based on MAD dispersion measure. The proposed ranking method based on AOR matches the ranking based on FR except for DSST tracker. According to the best scores assigned to each tracker, the ranking based on FR can be considered closer to the reference plot in Figure 6.3. However, the ranking based on FR assigns the same rank for STRUCK, KCF, and DSST trackers. Moreover, the ranking based on the individual AOR and FR do not discriminate among closely performing trackers.

Table 6.1: Scoring and ranking of the tested trackers based on AOR and FR individually.

AOR	STRUCK	KCF	SAMF	SCM	LOT	DSST	ASLA	Staple	TCNN	CCOT
Mean	0.448	0.477	0.541	0.431	0.33	0.528	0.435	0.586	0.648	0.676
Mean Rank	3	3	2	4	5	2	4	2	1	1
# Best scores	19	13	27	13	4	30	18	46	53	66
# 2nd best scores	16	31	33	18	14	24	20	22	25	17
FR										
Mean	0.224	0.2	0.165	0.262	0.312	0.206	0.264	0.148	0.058	0.046
Mean Rank	3	3	2	4	5	3	4	2	1	1
# Best scores	57	56	68	47	33	61	46	71	89	89
# 2nd best scores	11	15	14	13	30	13	19	11	8	7

6.7.3 IOF Ranking Results

Table 6.2 shows the scoring and ranking of the trackers based on IOF index according to MAD measure. As can be seen, the IOF ranking is 1. CCOT and TCNN, 2. STAPLE and SAMF, 3. DST and KCF, 4. STRUCK and SCM, and 5. LOT. This matches the ranking of the reference Figure 6.3. According to the number of best scores, as expected, the LOT tracker has been assigned the worst score, while CCOT is assigned as the highest scored tracker.

Table 6.2: IOF Scoring and ranking based on IOF index according to MAD measure.

IOF	STRUCK	KCF	SAMF	SCM	LOT	DSST	ASLA	Staple	TCNN	CCOT
Mean	0.41	0.441	0.509	0.388	0.275	0.488	0.391	0.555	0.632	0.666
Mean Rank	4	3	2	4	5	3	4	2	1	1
# Best scores	20	12	28	15	5	31	18	47	53	68
# 2nd best scores	18	30	31	15	13	23	21	18	29	14
Average FPS	11.61	38.59	15.22	0.587	1.089	58.81	7.15	55.25	0.446	0.581

Figure 6.4 shows how the IOF Rank-score plot perfectly matches the ranking results achieved using our proposed IOF measure as per table 6.2.

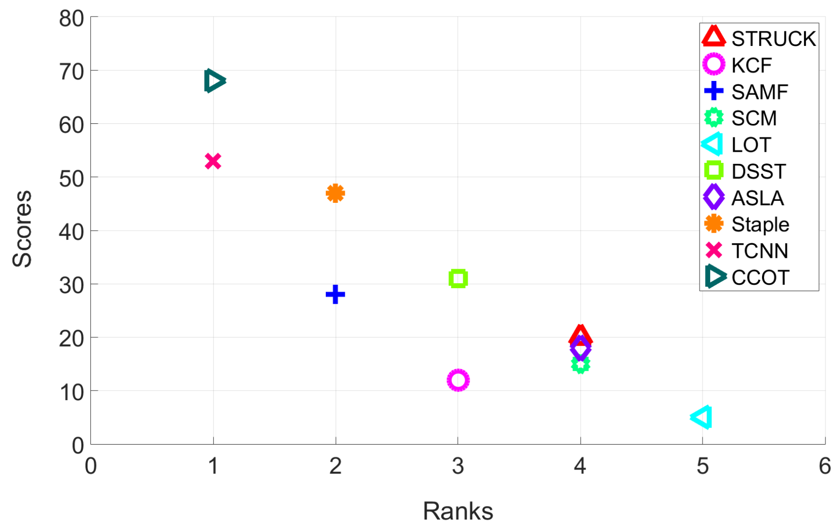


Figure 6.4: IOF Rank-score plot.

Figure 6.5 plots the IOF measure for all trackers; it does not only illustrate the ranking of the tested trackers, but also their relative quality. In addition, the quality of any tracker can be easily inferred relative to the best one. For example, the quality of the DSST is 0.958% that of SAMF.

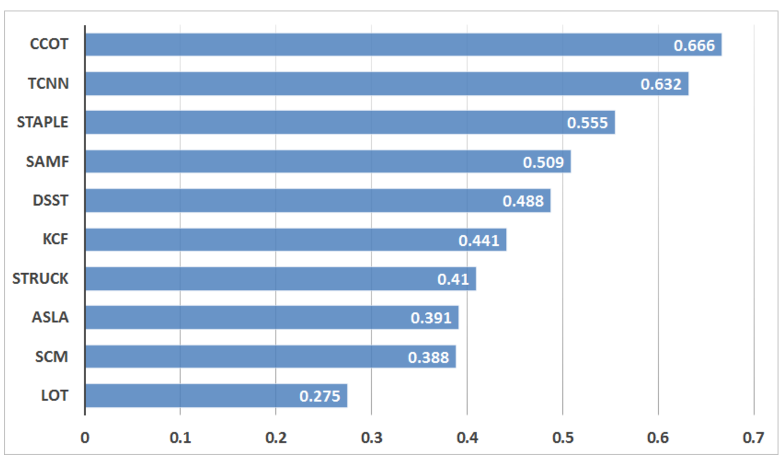


Figure 6.5: Ranking using IOF index.

Table 6.3 shows the IOF ranking for individual test sequences including the deviation thresholds (right two columns) that are used in the ranking process.

Table 6.3: IOF Scoring and ranking of the tested trackers for individual test sequences.

	STRUCK	KCF	SAMF	SCM	LOT	DSST	ASLA	STAPLE	TCNN	CCOT	Deviation threshold 1	Deviation threshold 2
Basketball	0.018	0.676	0.736	0.008	0.319	0.578	0.127	0.634	0.669	0.765	0.143	0.119
Biker	0.127	0.115	0.106	0.186	0.133	0.127	0.186	0.119	0.248	0.510	0.019	0.011
Bird1	0.023	0.007	0.094	0.024	0.006	0.032	0.003	0.177	0.059	0.057	0.023	0.017
Bird2	0.592	0.569	0.360	0.696	0.041	0.301	0.481	0.781	0.784	0.816	0.202	0.134
BlurBody	0.699	0.648	0.664	0.171	0.246	0.362	0.103	0.727	0.784	0.702	0.099	0.166
BlurCar1	0.796	0.797	0.802	0.032	0.501	0.765	0.241	0.373	0.774	0.859	0.061	0.163
BlurCar2	0.731	0.759	0.869	0.316	0.455	0.904	0.191	0.890	0.862	0.894	0.081	0.263
BlurCar3	0.820	0.568	0.581	0.250	0.124	0.841	0.023	0.831	0.817	0.832	0.132	0.226
BlurCar4	0.842	0.680	0.686	0.324	0.120	0.897	0.229	0.888	0.837	0.891	0.128	0.203
BlurFace	0.522	0.836	0.870	0.226	0.363	0.872	0.291	0.849	0.844	0.859	0.030	0.136
BlurOwl	0.761	0.046	0.589	0.213	0.182	0.045	0.107	0.254	0.822	0.836	0.188	0.075
Board	0.651	0.631	0.777	0.683	0.151	0.683	0.301	0.574	0.729	0.762	0.077	0.051
Bolt	0.000	0.678	0.720	0.000	0.549	0.723	0.000	0.789	0.679	0.639	0.087	0.129
Bolt2	0.069	0.000	0.000	0.000	0.308	0.000	0.000	0.681	0.375	0.527	0.034	0.000
Box	0.140	0.171	0.724	0.149	0.057	0.161	0.132	0.175	0.711	0.701	0.030	0.016
Boy	0.771	0.777	0.761	0.187	0.465	0.839	0.164	0.819	0.798	0.861	0.054	0.036
Car1	0.052	0.106	0.468	0.818	0.045	0.63	0.588	0.659	0.698	0.847	0.175	0.191
Car2	0.687	0.683	0.059	0.927	0.037	0.912	0.866	0.858	0.861	0.903	0.059	0.174
Car4	0.488	0.483	0.778	0.795	0.002	0.896	0.837	0.873	0.77	0.860	0.080	0.147
Car24	0.181	0.426	0.491	0.879	0.120	0.467	0.812	0.432	0.818	0.899	0.315	0.049
CarDark	0.892	0.614	0.729	0.872	0.316	0.845	0.887	0.871	0.722	0.85	0.042	0.107
CarScale	0.411	0.419	0.580	0.627	0.170	0.743	0.661	0.780	0.664	0.684	0.081	0.070
ClifBar	0.103	0.139	0.360	0.163	0.157	0.653	0.278	0.341	0.470	0.573	0.156	0.100
Coke	0.676	0.521	0.598	0.125	0.042	0.573	0.109	0.569	0.554	0.543	0.038	0.029
Couple	0.365	0.067	0.264	0.014	0.299	0.011	0.008	0.367	0.526	0.615	0.229	0.153
Coupon	0.869	0.944	0.935	0.900	0.161	0.898	0.906	0.899	0.153	0.894	0.018	0.007
Crossing	0.123	0.710	0.755	0.800	0.118	0.787	0.808	0.776	0.723	0.777	0.038	0.045
Crowds	0.011	0.794	0.727	0.596	0.000	0.732	0.708	0.802	0.693	0.728	0.050	0.029
Dancer	0.634	0.646	0.730	0.715	0.746	0.771	0.743	0.776	0.744	0.781	0.028	0.014
Dancer2	0.750	0.773	0.767	0.767	0.706	0.771	0.773	0.784	0.788	0.726	0.009	0.006
David	0.22	0.538	0.317	0.723	0.230	0.823	0.660	0.794	0.773	0.836	0.137	0.097
David2	0.856	0.827	0.805	0.778	0.609	0.810	0.851	0.788	0.757	0.786	0.024	0.020
David3	0.111	0.772	0.786	0.120	0.671	0.301	0.288	0.778	0.737	0.769	0.077	0.167
Deer	0.739	0.526	0.645	0.036	0.133	0.588	0.003	0.577	0.684	0.821	0.129	0.112
Diving	0.173	0.263	0.107	0.176	0.195	0.106	0.099	0.121	0.408	0.084	0.044	0.037
Dog	0.349	0.350	0.486	0.433	0.415	0.540	0.510	0.528	0.568	0.654	0.067	0.053
Dog1	0.546	0.551	0.705	0.700	0.595	0.757	0.713	0.817	0.809	0.799	0.094	0.061
Doll	0.229	0.534	0.632	0.819	0.672	0.847	0.604	0.835	0.853	0.840	0.104	0.067
DragonBaby	0.232	0.218	0.473	0.132	0.449	0.007	0.184	0.470	0.628	0.711	0.144	0.155
Dudek	0.712	0.728	0.795	0.770	0.448	0.788	0.760	0.656	0.843	0.809	0.040	0.037
FaceOcc1	0.725	0.753	0.795	0.800	0.409	0.765	0.416	0.793	0.672	0.795	0.035	0.081
FaceOcc2	0.785	0.751	0.746	0.684	0.488	0.780	0.715	0.762	0.727	0.772	0.027	0.026
Fish	0.857	0.839	0.825	0.681	0.124	0.803	0.834	0.783	0.829	0.863	0.026	0.025
FleetFace	0.522	0.589	0.635	0.632	0.554	0.634	0.608	0.658	0.733	0.679	0.034	0.025
Football	0.394	0.507	0.556	0.615	0.637	0.502	0.581	0.545	0.027	0.673	0.056	0.041
Football1	0.32	0.700	0.682	0.393	0.582	0.510	0.521	0.717	0.521	0.521	0.094	0.010
Freeman1	0.365	0.119	0.130	0.209	0.010	0.136	0.195	0.657	0.634	0.454	0.122	0.044
Freeman3	0.199	0.296	0.267	0.757	0.121	0.307	0.813	0.309	0.761	0.819	0.148	0.026
Freeman4	0.062	0.096	0.395	0.257	0.054	0.436	0.587	0.290	0.282	0.458	0.161	0.137
Girl	0.746	0.495	0.760	0.554	0.254	0.413	0.468	0.453	0.742	0.748	0.164	0.041
Girl2	0.057	0.004	0.005	0.082	0.508	0.039	0.035	0.041	0.707	0.541	0.038	0.035
Gym	0.473	0.379	0.499	0.028	0.291	0.253	0.413	0.337	0.466	0.400	0.080	0.063
Human2	0.676	0.154	0.652	0.143	0.216	0.244	0.316	0.732	0.741	0.766	0.252	0.061
Human3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.566	0.000	0.000
Human4	0.107	0.199	0.649	0.022	0.060	0.598	0.025	0.638	0.183	0.530	0.167	0.060
Human5	0.357	0.058	0.045	0.495	0.206	0.048	0.762	0.486	0.748	0.680	0.292	0.013

Human6	0.101	0.082	0.437	0.166	0.173	0.184	0.276	0.805	0.741	0.813	0.138	0.072
Human7	0.482	0.133	0.148	0.197	0.204	0.175	0.223	0.810	0.8	0.829	0.072	0.025
Human8	0.039	0.510	0.543	0.086	0.697	0.808	0.018	0.768	0.693	0.755	0.143	0.068
Human9	0.015	0.393	0.236	0.128	0.128	0.319	0.202	0.454	0.661	0.867	0.149	0.107
Ironman	0.012	0.039	0.049	0.039	0.027	0.027	0.039	0.020	0.209	0.543	0.011	0.011
Jogging_1	0.106	0.047	0.786	0.042	0.021	0.045	0.045	0.042	0.734	0.786	0.014	0.003
Jogging_2	0.026	0.022	0.774	0.145	0.025	0.027	0.027	0.028	0.701	0.697	0.004	0.002
Jump	0.064	0.036	0.027	0.040	0.130	0.024	0.035	0.005	0.338	0.139	0.019	0.012
Jumping	0.634	0.218	0.201	0.116	0.616	0.136	0.067	0.199	0.740	0.683	0.117	0.063
KiteSurf	0.141	0.373	0.164	0.121	0.002	0.134	0.152	0.742	0.624	0.119	0.026	0.021
Lemming	0.407	0.265	0.767	0.035	0.566	0.216	0.036	0.082	0.701	0.786	0.276	0.113
Liquor	0.200	0.276	0.402	0.163	0.768	0.289	0.357	0.651	0.815	0.770	0.198	0.078
Man	0.879	0.831	0.846	0.854	0.047	0.842	0.860	0.852	0.830	0.811	0.013	0.011
Matrix	0.017	0.034	0.115	0.052	0.107	0.032	0.120	0.099	0.355	0.559	0.059	0.047
Mhyang	0.817	0.796	0.866	0.818	0.105	0.806	0.926	0.779	0.793	0.851	0.025	0.012
MotorRolling	0.122	0.021	0.018	0.036	0.057	0.018	0.043	0.025	0.637	0.019	0.012	0.007
MountainBike	0.710	0.711	0.711	0.642	0.533	0.730	0.749	0.717	0.753	0.654	0.028	0.006
Panda	0.524	0.072	0.180	0.434	0.016	0.056	0.563	0.193	0.565	0.288	0.189	0.068
RedTeam	0.488	0.500	0.657	0.521	0.028	0.564	0.659	0.567	0.585	0.706	0.071	0.042
Rubik	0.472	0.613	0.656	0.545	0.261	0.648	0.691	0.635	0.625	0.784	0.043	0.030
Shaking	0.168	0.006	0.027	0.708	0.076	0.716	0.461	0.006	0.686	0.663	0.308	0.020
Singer1	0.358	0.354	0.531	0.859	0.044	0.824	0.816	0.822	0.732	0.873	0.091	0.172
Singer2	0.005	0.732	0.004	0.012	0.340	0.781	0.747	0.783	0.691	0.002	0.266	0.003
Skater	0.636	0.610	0.606	0.609	0.443	0.586	0.508	0.604	0.566	0.559	0.021	0.022
Skater2	0.431	0.566	0.564	0.351	0.556	0.518	0.496	0.425	0.628	0.621	0.062	0.064
Skating1	0.164	0.489	0.621	0.409	0.131	0.527	0.388	0.261	0.369	0.293	0.113	0.073
Skating2_1	0.156	0.394	0.173	0.004	0.127	0.328	0.048	0.474	0.537	0.519	0.173	0.062
Skating2_2	0.380	0.328	0.426	0.049	0.398	0.056	0.048	0.199	0.481	0.399	0.099	0.079
Skiing	0.001	0.005	0.005	0.012	0.002	0.008	0.011	0.020	0.491	0.422	0.006	0.003
Soccer	0.122	0.415	0.056	0.237	0.232	0.411	0.180	0.082	0.463	0.633	0.164	0.150
Subway	0.651	0.754	0.757	0.696	0.570	0.047	0.726	0.743	0.665	0.701	0.046	0.040
Surfer	0.413	0.431	0.678	0.038	0.269	0.274	0.021	0.106	0.746	0.719	0.271	0.162
Suv	0.401	0.883	0.858	0.503	0.587	0.811	0.749	0.833	0.418	0.790	0.100	0.084
Sylvester	0.73	0.593	0.210	0.655	0.557	0.583	0.532	0.560	0.761	0.624	0.046	0.030
Tiger1	0.631	0.785	0.618	0.216	0.074	0.604	0.140	0.764	0.672	0.676	0.095	0.062
Tiger2	0.581	0.26	0.536	0.127	0.063	0.287	0.058	0.684	0.601	0.584	0.180	0.069
Toy	0.437	0.475	0.6	0.318	0.199	0.702	0.467	0.649	0.686	0.649	0.111	0.038
Trans	0.526	0.384	0.537	0.522	0.468	0.461	0.557	0.551	0.586	0.522	0.030	0.022
Trellis	0.551	0.631	0.843	0.723	0.210	0.773	0.567	0.845	0.767	0.742	0.106	0.063
Twinnings	0.573	0.564	0.703	0.508	0.491	0.787	0.581	0.763	0.621	0.793	0.097	0.032
Vase	0.317	0.316	0.448	0.333	0.371	0.541	0.349	0.580	0.566	0.516	0.092	0.024
Walking	0.571	0.530	0.711	0.725	0.695	0.744	0.778	0.741	0.712	0.683	0.029	0.028
Walking2	0.510	0.395	0.19	0.816	0.169	0.800	0.177	0.777	0.234	0.799	0.279	0.038
Woman	0.736	0.662	0.633	0.637	0.016	0.651	0.036	0.749	0.738	0.740	0.080	0.021
Mean	0.409	0.441	0.509	0.387	0.275	0.487	0.391	0.554	0.631	0.665		
Mean Rank	4	3	2	4	5	3	4	2	1	1		
# Best scores	20	12	28	15	5	31	18	47	53	68		
# 2nd best	18	30	31	15	13	23	21	18	29	14		

6.7.4 Recovery and Drift Results

Table 6.4 shows the ranking of all trackers based on recovery measure according to MAD dispersion measure. The assigned ranks based on recovery exactly matches the ranks given in the reference Figure 6.3. Table 6.5 shows the ranking of all trackers based on the drift metric according to MAD measure. The assigned ranks based on the drifts metric exactly match the ranks given by reference ranking of Figure 6.3. In table 6.6, we give the recover-to-drift measure $pRD = \frac{(R-D)}{(L-100)}$ where R is the sum of recoveries over all 100 videos, D is the sum of drifts over all videos, and L is the total number of frames of the videos. Note that we subtract 100 as we skip the first frame in each video. As can be seen, proposed pRD finely distinguishes the 10 tested trackers in term of pure drift (or pure recovery). It perfectly matches the ranking in the reference figure 6.3, that is: 1. CCOT, 2. TCNN, 3. STAPLE, 4. SAMF, 5. DSST, 6. KCF, 7. STRUCK, 8. SCM, 9. ASLA, 10. LOT. We notice how the LOT tracker has negative pRD value since (as per tables 6.4 and 6.5) it has higher drift than recovery.

Table 6.4: Average recovery-based ranking over all 100 videos.

Recovery	STRUCK	KCF	SAMF	SCM	LOT	DSST	ASLA	Staple	TCNN	CCOT
Mean	291.89	330.2	383	298.8	169.3	339	273	399.6	473	455
Mean Rank	4	3	2	3	5	3	4	2	1	1

Table 6.5: Average drift-based ranking over all 100 videos.

Drifts	STRUCK	KCF	SAMF	SCM	LOT	DSST	ASLA	Staple	TCNN	CCOT
Mean	122.31	108.7	84.4	130.3	175	102	122	76.52	34.1	14.6
Mean Rank	4	3	2	4	5	3	4	2	1	1

Table 6.6: pRD-based ranking over all 100 videos.

pRD	STRUCK	KCF	SAMF	SCM	LOT	DSST	ASLA	Staple	TCNN	CCOT
Mean	0.2877	0.376	0.506	0.286	-0.01	0.403	0.255	0.548	0.745	0.748
Mean Rank	7	6	4	8	10	5	9	3	2	1

For more investigation, we created a plot that shows the relation between the R vs. D for each tracker as shown in Figure 6.6. This plot presents detailed information about trackers quality and well agrees with the reference measure given in Figure 6.3. It well distinguishes the ten trackers as follows: 1. CCOT and TCNN, 2. STAPLE, 3. SAMF, 4. DSST, 5. KCF, 6. STRUCK and SCM, 7. ASLA, 8. LOT. The plot clusters trackers into five separated groups (separated by dashed lines) which facilitate the ranking process based on tracking characteristics such as the number of drifts and recoveries of each tracker.

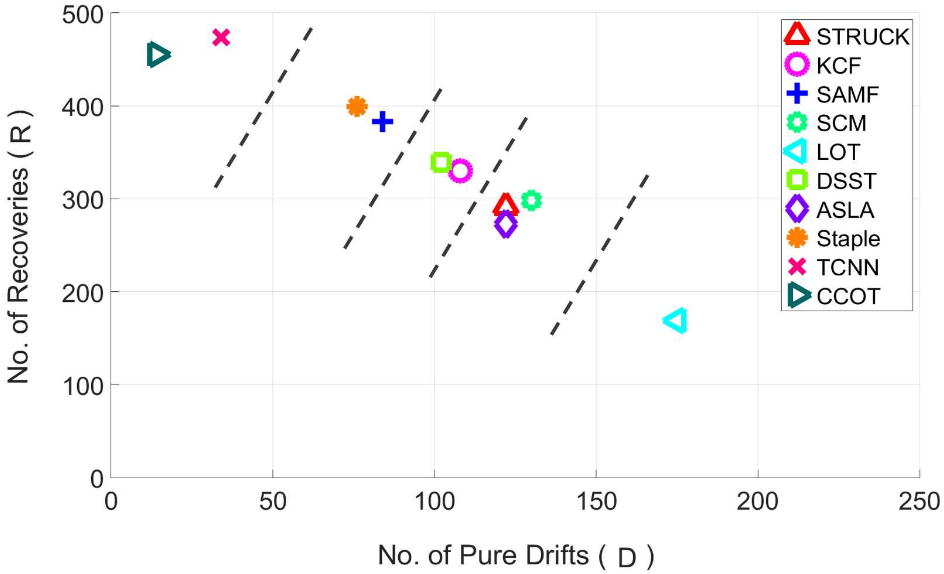


Figure 6.6: Recovery (R) vs. drift (D) plot.

6.7.5 Stability of the Proposed Framework

To test the stability of our framework, we selected the trackers such that some of which have similar performance, and the others have fully different performance. Comparing Tables 6.1 and 6.2, we can see that using the unified metric IOF, instead of individual metrics, trackers are less scattered. In addition, IOF shows to discriminate between closely performing trackers better than AOR and FR when used individually. This is also shown in the IOF scatter plot of Figure 6.4 for the ranks and scores of the tested trackers. As seen, both CCOT and TCNN trackers show the best quality (highest rank and scores) at the top left corner, while LOT tracker shows the worst quality (lowest rank and scores) at the bottom right corner. IOF shows that the top ranked trackers (same rank) also have the highest scores. Since IOF is a simple and effective metric, we thus propose to use the IOF index as a unified metric for tracker evaluation.

6.7.6 Speed Ranking

The speed of all evaluated trackers, in FPS, is shown in Table 6.2. The tracking algorithms are run on Intel Xeon(R) 3.6 GHz PC with 16 GB memory; all but STRUCK (C/C++) are coded in Matlab. As can be seen, the TCNN and CCOT trackers, ranked the first, have the lowest FPS (0.446 and 0.581), while DSST and STAPLE trackers have the best FPS (58.81 and 55.25) among all trackers. Combining the rate, score, and rank for all trackers, we can obviously conclude that the best of all tested trackers is STAPLE.

6.8 Conclusion

In this chapter, a framework is presented for effective scoring and ranking of trackers using known as well as three proposed performance measures coupled with a statistical dispersion metric (MAD) to account for the presence of outliers. The framework is verified on ten state-of-the-art different performing trackers from view points of accuracy and performance. Results on publicly available data set of 100 test sequences showed that the scoring process provides detailed information about trackers' quality that complements the ranking. The proposed index measure (combining overlap and failure rate) aims to measure the quality of trackers using a unified metric to facilitate ranking and scoring. The proposed index is shown to better replace the individual overlap and failure rate metrics. The proposed recovery and pure drift metrics facilitate the ranking process and support the results of the unified index.

For future work, we plan to extend the proposed framework to use multi-objective optimization techniques such as artificial immune systems for ranking to reflect the different aspects of the tracking performance.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Object tracking is a sophisticated process inevitably causing frequent tracking drift or even failure as a result of various challenges in the tracking environment. Adaptive appearance modeling has attracted significant attention as it accounts for drastic appearance changes. The output of such trackers is a bounding box representation, the center of which is considered as the estimated object location. Such bounding box may not provide accurate foreground/background discrimination and may not be centered correctly around the target object, which affects the accuracy of the overall tracking process.

This work has investigated approaches for enhancing the quality and facilitating the evaluation of tracking algorithms. First, an automatic drift detection approach, using saliency features of the target object, integrated with a drift correction mechanism through an automatic seeded segmentation, was investigated. The proposed approach was evaluated on a publicly available dataset of 100 video sequences that cover 11 different tracking challenges, using five recent trackers that have different attributes from points of view of tracking accuracy, performance, and methodology. It was found that the proposed approach successfully tracks the target object under different challenge conditions, while the base trackers drifted away from the target object.

The second approach for enhancing the tracking quality was the investigation of using the clonal selection-based artificial immune system optimization for the selection of the best performing configurations of support vector machines and object segmentation. The obtained results show that the proposed method applied to STRUCK tracker has led to an enhanced quality that outperforms the original algorithm according to various objective measures. In addition, the selection of the parameters of graph cut segmentation using artificial immune system optimization was explored. A better segmentation quality has been achieved according to different objective measures. The proposed method overcomes the bias of human intervention and the tedious work adopted for parameter selection, while achieving global near optimal configurations that in turn lead to better tracking quality. It can be concluded that the artificial immune system optimization can effectively serve as an attractive tool for computer vision applications such as segmentation and tracking.

For facilitating the evaluation of object tracking algorithms, this work introduced a framework for scoring and ranking of object trackers, using known quality metrics, coupled with a robust estimator against dispersion and outliers. Three new performance measures were proposed to facilitate trackers' evaluation: A simple and straight forward performance index, that facilitates the ranking and scoring process, was proposed. Two other measures; recovery and pure drift measures, were also introduced. Ten state-of-the-art different performing trackers are scored and ranked using the proposed framework on a public benchmark of 100 video sequences. The obtained results show that the proposed framework facilitates the evaluation of the relative performance of different trackers. The proposed unified index does not only illustrate the ranking of the tested trackers, but also their relative quality. The quality of any tracker can be easily inferred relative to the best one. The proposed recovery and pure drift metrics facilitate the ranking process and support the results of the unified index.

7.2 Future Work

In addition to the use of saliency features for automatic drift detection, we propose to investigate the application of different saliency models for drift detection, and study their effect on the tracking performance and accuracy.

G-Cut object segmentation is a computationally expensive method. We propose to investigate the use of fast object segmentation methods so to improve the speed of drift correction.

Recent approaches in object detection incorporate the use of objectness measures. We propose to investigate the use of objectness for automatic drift detection (preliminary results showed a promising improvement of the tracking quality using objectness measures (Appendix B)). An interesting future work could include the integration of both saliency features and objectness measures in a unified framework for automatic drift detection.

To enhance the performance of the proposed AIS tracking method, we propose to enhance the artificial immune system model to achieve faster convergence rate. In addition, it is intended to study the effect of using different features for the online training of the complementary support vector machine model on the tracking accuracy.

For the proposed ranking framework, we plan to investigate using multi-objective optimization techniques such as artificial immune systems for ranking to reflect the different aspects of the tracking quality.

References

[1] L. N. D. Castro and F. J. V. Zuben, "The clonal selection algorithm with engineering applications," in *GECCO Workshop Proceedings*, July 2000, pp. 36–37.

[2] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *Proc. IEEE Int. Conf. Computer Vision*, Barcelona, Nov. 2011, pp. 263–270.

[3] L. Bertinetto et al., "Staple: Complementary learners for real-time tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Oct. 2016, pp. 1401–1409.

[4] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, pp. 1–45, 2006.

[5] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A survey of appearance models in visual object tracking," *ACM Trans. Intell. Syst. Technol.*, vol. 4, pp. 1–48, 2013.

[6] H. Yang, L. Shao, F. Zheng, L. Wang, and Z. Song, "Recent advances and trends in visual tracking: A review," *J. Neurocomputing*, vol. 74, pp. 3823–3831, 2011.

[7] H. Chi, S. Kang, and X. Wang, "Research trends and opportunities of augmented reality applications in architecture, engineering, and construction," *Autom. Construction*, vol. 33, pp. 116–122, 2013.

[8] X. Jia, H. Lu, and M. H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Providence, Rhode Island, June 2012, pp. 1822–1829.

[9] W. Zhong, H. Lu, and M. H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Washington, DC, USA, June 2012, pp. 1838–1845.

[10] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 37, pp. 583–596, 2015.

[11] L. Yang and Z. Jianke, "A scale adaptive kernel correlation filter tracker with feature integration," in *Proc. European Conf. Computer Vision*, Cham, Sept. 1-5 2014, pp. 254–265, Springer.

[12] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," *Int. J. Computer Vision*, vol. 111, pp. 213–228, 2015.

[13] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conf.*, Nottingham, Sept. 1-5 2014, pp. 1838–1845, BMVA Press.

[14] S. Lin and S. Chen, "Parameter tuning, feature selection and weight assignment of features for case-based reasoning by artificial immune system," *J. Applied Soft Computing*, vol. 11, pp. 5042–5052, 2011.

[15] I. Aydin, M. Karakose, and E. Akin, "A multi-objective artificial immune algorithm for parameter optimization in support vector machine," *J. Applied Soft Computing*, vol. 11, pp. 120–129, 2011.

[16] Q. Wang, F. Chen, W. Xu, and Ming-Hsuan Yang, "An experimental comparison of online object-tracking algorithms," in *SPIE Optical Engineering+ Applications*, California, USA, Sept. 2011, pp. 81381A–81381A.

[17] S. Hare et al., "Struck: Structured output tracking with kernels," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 38, pp. 2096–2109, 2016.

[18] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2016, pp. 4293–4302.

[19] C. Yang et al., "Efficient mean-shift tracking via a new similarity measure," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2005, pp. 176–183.

[20] P. Pérez et al., "Color-based probabilistic tracking," *Proc. European Conf. Computer Vision*, vol. 5, pp. 661–675, 2002.

[21] D. Ma, Z. Yu, J. Yu, and W. Pang, "A novel object tracking algorithm based on compressed sensing and entropy of information," *Mathematical Problems Engineering*, 2015.

[22] K. Zhang, L. Zhang, and M. Yang, "Real-time compressive tracking," in *Proc. European Conf. Computer Vision*, 2012, pp. 864–877.

[23] K.E. Papoutsakis and A. Argyros, "Integrating tracking with fine object segmentation," *Image Vis. Comput.*, vol. 31, pp. 771–785, 2013.

[24] K.E. Papoutsakis and A. Argyros, "Object tracking and segmentation in a closed loop," in *Int. Conf. Adv. Visual Comput. (ISCV)*, Las Vegas, NV, USA, June 2010, pp. 405–416, Springer-Verlag.

[25] Y. Wang, G. Jiang, and C. Jiang, "Mean shift tracking with graph cuts based image segmentation," in *Int. Congress Image Signal Processing (CISP)*, Chongqing, Sichuan, China, Oct. 2012, pp. 675–679.

[26] D. Freedman and M. Turek, "Illumination-invariant tracking via graph cuts," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, San Diego, CA, USA, June 2005, pp. 10–17.

[27] H. Wang, N. Sang, and Y. Yan, "Real-time tracking combined with object segmentation," in *Proc. IEEE Int. Conf. Pattern Recognition*, Washington, DC, USA, 2014, pp. 4098–4103.

[28] J. Son, I. Jung, K. Park, and B. Han, "Tracking-by-segmentation with online gradient boosting decision tree," in *Proc. IEEE Int. Conf. Computer Vision*, 2015, pp. 3056–3064.

[29] B. Vasileios, F. Schubert, N. Navab, and S. Ilic, "Segmentation based particle filtering for real-time 2D object tracking," in *Proc. European Conf. Computer Vision*, Florence, Italy, 2012, pp. 842–855, Springer-Verlag.

[30] P. Chockalingam, N. Pradeep, and S. Birchfield, "Adaptive fragments-based tracking of non-rigid objects using level sets," in *Proc. IEEE Int. Conf. Computer Vision*, Kyoto, Japan, Sept. 2009, pp. 1530–1537.

[31] B. Babenko, M. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 33, pp. 1619–1632, 2011.

[32] G.M. Rao and C. Satyanarayana, "Visual object target tracking using particle filter: a survey," *Int. J. Image Graphics Signal Processing*, vol. 5, pp. 1250, 2013.

[33] Y. Huang, T. S. Huang, and H. Niemann, "Segmentation-based object tracking using image warping and kalman filtering," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2002, pp. 601–604.

[34] I. Kyriakides, D. Morrell, and A. Papandreou-Suppappola, "A particle filtering approach to constrained motion estimation in tracking multiple targets," in *Asilomar Conf. Signals, Systems and Computers*, Beijing, China, 94–98 Oct. 2005.

[35] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Beyond sliding windows: Object localization by efficient subwindow search," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2008, pp. 1–8.

[36] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 32, pp. 1627–1645, 2010.

[37] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, "Fast, accurate detection of 100,000 object classes on a single machine," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2013, pp. 1814–1821.

[38] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 34, pp. 2189–2202, 2012.

[39] A. Borji, M. Cheng, H. Jiang, and J. Li, "Salient object detection: A survey," *arXiv preprint arXiv:1411.5878*, 2014.

[40] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *Int. J. Computer Vision*, vol. 60, pp. 63–86, 2004.

[41] L. Marchesotti, C. Cifarelli, and G. Csurka, "A framework for visual saliency detection with applications to image thumbnailing," in *Proc. IEEE Int. Conf. Computer Vision*, Sept. 2009, pp. 2232–2239.

[42] R. Achanta, F. Estrada, P. Wils, and S. Süsstrunk, "Salient region detection and segmentation," in *Proc. IEEE Int. Conf. Computer Vision Systems*, Oct. 2008, pp. 66–75.

[43] R. Achanta, F. Estrada, P. Wils, and S. Süsstrunk, "Frequency-tuned salient region detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2009, pp. 1597–1604.

[44] M. Cheng, G. Zhang, N. J. Mitra, X. Huang, and S. Hu, "Global contrast based salient region detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2011, pp. 409–416.

[45] C. Li, C. Xu, C. Gui, and M. D. Fox, "A saliency map based on sampling an image into random rectangular regions of interest," *J. Pattern Recognition*, vol. 45, pp. 3114–3124, 2012.

[46] H. Possegger, T. Mauthner, and H. Bischof, "In defense of color-based model-free tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2015, pp. 2113–2120.

[47] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, pp. 679–698, 1986.

[48] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Proc. European Conf. Computer Vision*, pp. 430–443, 2006.

[49] T. Tuytelaars et al., "Local invariant feature detectors: a survey," *Foundations and trends® in computer graphics and vision*, vol. 3, pp. 177–280, 2008.

[50] B. KP Horn and B.G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, pp. 185–203, 1981.

[51] D.G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 1999, pp. 1150–1157.

[52] H. Bay et al., "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol. 110, pp. 346–359, 2008.

[53] E. Rublee, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Computer Vision*, 2011, pp. 2564–2571.

[54] E. Maggio and A. Cavallaro, *Video tracking: theory and practice, 1st Edition*, John Wiley & Sons, 2011.

[55] L. Čehovin, A. Leonardis, and M. Kristan, "Visual object tracking performance measures revisited," *Proc. IEEE Int. Conf. Image Processing (ICIP)*, vol. 25, pp. 1261–1274, 2016.

[56] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Portland, OR, USA, June 2013, pp. 2411–2418.

[57] Y. Wu, J. Lim, and M. H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 37, pp. 1834–1848, 2015.

[58] J. E. Hunt and D. E. Cooke, "Learning using an artificial immune system," *J. Network Comp. Applications*, vol. 19, pp. 189–212, 1996.

[59] L.N. De Castro and F.J.V. Zuben, "Learning and optimization using the clonal selection principle," *IEEE Trans. Evolutionary Computation*, vol. 6, pp. 239–251, 2002.

[60] E. Talbi, *Metaheuristics: From Design to Implementation*, Wiley Publishing, Hoboken, New Jersey, 2009.

[61] W. Wu, S. Lin, and W. K. Moon, "An artificial immune system-based support vector machine approach for classifying ultrasound breast tumor images," *J. Digital Imaging*, vol. 28, pp. 576–585, 2015.

[62] D.F. McCoy and V. Devarajan, "Artificial immune systems and aerial image segmentation," in *Computational Cybernetics Simulation*, June 1997, pp. 867–872.

[63] E. Cuevas et al., "Multithreshold segmentation based on artificial immune systems," *Mathematical Problems in Engineering*, vol. 874761, 2012.

[64] X. Yu and M. Gen, *Introduction to evolutionary algorithms*, Springer Science & Business Media, 2010.

[65] Leandro Nunes De Castro, Fernando J Von Zuben, and Getúlio A de Deus Jr, "The construction of a boolean competitive neural network using ideas from immunology," *Neurocomputing*, pp. 51–85, 2003.

[66] I. Matthews, T. Ishikawa, and S. Baker, "The template update problem," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, pp. 810–815, 2004.

[67] D. Schreiber, "Robust template tracking with drift correction," *J. Applied Soft Computing*, vol. 28, pp. 1483–1491, 2007.

[68] B. C. Ko and J. Nam, "Object-of-interest image segmentation based on human attention and semantic region clustering," *J. Opt. Soc. Am. A*, vol. 23, pp. 2462–2470, 2006.

[69] J. Han et al., "Unsupervised extraction of visual attention objects in color images," *IEEE Trans. Circuits Sys. Video Technology*, vol. 16, pp. 141–145, 2006.

[70] X. Liao et al., "Automatic image segmentation using salient key point extraction and star shape prior," *Signal Processing*, vol. 105, pp. 122–136, 2014.

[71] U. Rutishauser et al., "Is bottom-up attention useful for object recognition?," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2004, pp. II–37–II–44 Vol.2.

[72] T. Chen et al., "Sketch2photo: Internet image montage," *ACM Transactions on Graphics (TOG)*, vol. 28, pp. 124:1–124:10, 2009.

[73] G. Pezzulo et al., *The challenge of anticipation: A unifying framework for the analysis and design of artificial cognitive systems*, vol. 5225, Springer, 2008.

[74] L. Wen et al., "JOTS: Joint online tracking and segmentation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June. 2015, pp. 2226–2234.

[75] Y. Boykov and M-P Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in ND images," in *Proc. IEEE Int. Conf. Computer Vision.* July 2001, pp. 105–112, Springer.

[76] L. Grady, "GrabCut: Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, pp. 309–314, 2004.

[77] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 28, pp. 1768–1783, 2006.

[78] M. Schmidt and K. Alahari, "Generalized fast approximate energy minimization via graph cuts: Alpha-expansion beta-shrink moves," *27th Conf. Uncertainty Artificial Intelligence (UAI)*, 2011.

[79] A. Suga et al., "Object recognition and segmentation using SIFT and Graph Cuts," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Oct. 2008, pp. 1–4.

[80] J. Shan, H. Cheng, and Y. Wang, "A novel automatic seed point selection algorithm for breast ultrasound images," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Oct. 2008, pp. 1–4.

[81] S. Gao and J. Yang, "Saliency-seeded localizing region-based active contour for automatic natural object segmentation," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2012, pp. 3644–3647.

[82] X. Yuan, F. Xiang, and Z. Wang, "Automatic color image segmentation using CSIFT and Graph Cuts," in *Information Science and Technology (ICIST)*, 2013, pp. 365–368.

[83] K. Kang et al., "T-CNN: Tubelets with convolutional neural networks for object detection from videos," *arXiv preprint arXiv:1604.02532*, pp. 1 – 10, 2016.

[84] M. Danelljan et al., "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proc. European Conf. Computer Vision*, 2016, pp. 472–488.

[85] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proc. IEEE Int. Conf. Computer Vision*, 2015, pp. 3119–3127.

[86] D. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2010, pp. 2544–2550.

[87] X. Feng, W. Mei, and D. Hu, "A review of visual tracking with deep learning," *Advances Intell. System Research (AIIE)*, vol. 133, pp. 167–181, 2016.

[88] L. Elazary and L. Itti, "Interesting objects are visually salient," *J. vision*, vol. 8, pp. 1–15, 2008.

[89] P.F. Felzenszwalb and D.P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Computer Vision*, vol. 59, pp. 167–181, 2004.

[90] H.R. Kher and V.K. Thakar, "Scale invariant feature transform based image matching and registration," in *Proc. IEEE Int. Conf. Signal Image Processing (ICSIP)*, 2014, pp. 50–55.

[91] M. Kristan et al., "The visual object tracking VOT2013 challenge results," in *Proc. IEEE Int. Conf. Computer Vision Workshops*, Sydney, Dec. 2013, pp. 98–111.

[92] M. Kristan et al., "The visual object tracking VOT2014 challenge results," in *European Conf. Computer Vision Workshops*, Zurich, Switzerland, Sept. 2014, pp. 191–217, Springer.

[93] M. Kristan et al., "The visual object tracking VOT2015 challenge results," in *Proc. IEEE Int. Conf. Computer Vision Workshops*, Santiago, Dec. 2015, pp. 564–586.

[94] A. Li, M. Lin, Y. Wu, M.H. Yang, and S. Yan, "NUS-PRO: A new visual tracking challenge.," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 38, pp. 335, 2016.

[95] M. Kristan et al., "A novel performance evaluation methodology for single-target trackers," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 38, pp. 2137–2155, 2016.

[96] M. Kristan et al., "The visual object tracking VOT2016 challenge results," in *European Conf. Computer Vision Workshops*. Jan. 2016, pp. 777–823, Springer.

[97] D. Tsai, M. Flagg, and J. M.Rehg, "Motion coherent tracking with multi-label mrf optimization," *British Machine Vision Conf.*, 2010.

[98] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg, "Video segmentation by tracking many figure-ground segments," in *Proc. IEEE Int. Conf. Computer Vision Systems*, 2013, pp. 2192–2199.

[99] P. Kohli, A. Osokin, and S. Jegelka, "A principled deep random field model for image segmentation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2013, pp. 1971–1978.

[100] Y. Zhao, S. Zhu, and S. Luo, "Co3 for ultra-fast and accurate interactive segmentation," in *18th ACM Int. Conf. Multimedia*, 2010, pp. 93–102.

[101] J. Mille, S. Bougleux, and L.D. Cohen, "Combination of piecewise-geodesic paths for interactive segmentation," *Int. J. Computer Vision*, vol. 112, pp. 1–22, 2015.

[102] Y. Zhang, Y. Tang, B. Fang, and Z. Shang, "Real-time object tracking in video pictures based on self-organizing map and image segmentation," in *Proc. Int. Joint Conf. Inf. Technology Artificial Intell.*, Chongqing, China, Dec. 2014, pp. 559–563, IEEE.

[103] B. Babenko, M. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 33, pp. 1619–1632, 2010.

[104] S. Hare, A. Saffari, and P. H. S. Torr, "Efficient online structured output learning for keypoint-based object tracking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2012, pp. 1894–1901.

[105] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 34, pp. 1409–1422, 2012.

[106] F. Comaschi, S. Stuijk, T. Basten, and H. Corporaal, "Online multi-face detection and tracking using detector confidence and structured SVMs," in *12th IEEE Int. Conf. Advanced Video Signal Based Surveillance (AVSS)*, Aug. 2015, pp. 1–6.

[107] P. Chen, R. Fan, and C. Lin, "A study on SMO-type decomposition methods for support vector machines," *IEEE Trans. Neural Networks*, vol. 17, pp. 893–908, 2006.

[108] E. Osuna, R. Freund, and F. Girosit, "Training support vector machines: an application to face detection," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Sept. 1997, pp. 130–136.

[109] Y. Ren and F. Zhang, "Hand gesture recognition based on meb-svm," in *Int. Conf. Embedded Software Systems*, May 2009, pp. 344–349.

[110] S. Zhang, X. Yu, Y. Sui, S. Zhao, and L. Zhang, "Object tracking with multi-view support vector machines," *IEEE Trans. Multimedia*, vol. 17, pp. 265–278, 2015.

[111] S.S. Keerthi, "Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms," *IEEE Transactions on Neural Networks*, vol. 13, pp. 1225–1229, 2002.

[112] S. Lessmann, R. Stahlbock, and S.F. Crone, "Genetic algorithms for support vector machine model selection," in *IEEE Int. Joint Conf. Neural Network Proceedings*, July 2009, pp. 3063–3069.

[113] S. Yuan and F. Chu, "Fault diagnosis based on support vector machines with parameter optimisation by artificial immunisation algorithm," *Mechanical Systems and Signal Processing*, vol. 21, pp. 1318–1330, 2007.

[114] I. Aydin, M. Karakose, and E. Akin, "Artificial immune based support vector machine algorithm for fault diagnosis of induction motors," in *Int. Aegean Conf. Electrical Machines Power Electronics*, Sept. 2007, pp. 217–221.

[115] D. P. Chau et al., "Online parameter tuning for object tracking algorithms," *Image Vis. Comput.*, vol. 32, pp. 287–302, 2014.

[116] E.D. Ulker and S. Ulker, "Comparison study for clonal selection algorithm and genetic algorithm," *Int. J. Computer Science Information Technology (IJCSIT)*, vol. 4, pp. 107–118, 2012.

[117] W. Ma et al., "Medical image segmentation based on immune clonal optimization," in *Intelligent Computing and Intelligent Systems*, 2009, pp. 377–381.

[118] F. Yi and I. Moon, "Image segmentation: A survey of graph-cut methods," in *Int. Conf. Systems Informatics (ICSAI)*, 2012, pp. 1936–1941.

[119] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Mathematical Problems in Engineering*, vol. 4, pp. 321–331, 1988.

[120] S. Osher and J.A. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations," *Mathematical Problems in Engineering*, vol. 79, pp. 12–49, 1988.

[121] B. Peng and O. Veksler, "Parameter selection for graph cut based image segmentation," in *British Machine Vision Conf.*, 2008, pp. 42–44.

[122] K.O. Oyebode and J.R. Tapamo, "Adaptive parameter selection for graph cut-based segmentation on cell images," *Image Analysis & Stereology*, vol. 35, pp. 29–37, 2016.

[123] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, pp. 1124–1137, 2004.

[124] J. Pont-Tuset and F. Marques, "Measures and meta-measures for the supervised evaluation of image segmentation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June. 2013, pp. 2131–2138.

[125] C. Yang et al., "Saliency detection via graph-based manifold ranking," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2013, pp. 3166–3173.

[126] F.E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, pp. 1–21, 1969.

[127] F.R. Hampel, "A general qualitative definition of robustness," *The Annals of Mathematical Statistics*, pp. 1887–1896, 1971.

[128] P.L. Davies and U. Gather, "The breakdown point-examples and counterexamples," *J. REVSTAT Statistical*, vol. 5, pp. 1–17, 2007.

[129] F.R. Hampel, "The influence curve and its role in robust estimation," *J. American Statistical association*, vol. 69, pp. 383–393, 1974.

[130] C. Leys et al., "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *J. Experimental Social Psychology*, vol. 49, pp. 764–766, 2013.

[131] L. Čehovin, M. Kristan, and A. Leonardis, "Is my new tracker really better than yours?," in *IEEE Winter Conf. App. Computer Vision*, Co, USA, March 2014, pp. 540–547.

[132] Y. Pang and H. Ling, "Finding the best from the second bests - inhibiting subjective bias in evaluation of visual tracking algorithms," in *Proc. IEEE Int. Conf. Computer Vision*, Sydney, Australia, Dec. 2013, pp. 2784–2791.

[133] J.P. Simmons, L.D. Nelson, and U. Simonsohn, "False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant," *Psychological Science*, vol. 22, pp. 1359–1366, 2011.

[134] D. Ross et al., "Incremental learning for robust visual tracking," *Int. J. Computer Vision*, vol. 77, pp. 125–141, 2008.

[135] T. Nawaz and A. Cavallaro, "A protocol for evaluating video trackers under real-world conditions," *IEEE Trans. Image Process.*, vol. 22, pp. 1354–1361, 2013.

[136] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. European Conf. Computer Vision*, 2014, pp. 391–405.

[137] B. Adenso-Diaz and M. Laguna, "Fine-tuning of algorithms using fractional experimental designs and local search," *J. Oper. Res.*, vol. 54, pp. 99–114, 2006.

[138] J. Ferryman and A. Shahrokni, "Pets2009: Dataset and challenge," in *Twelfth IEEE Int. Workshop Performance Evaluation of Tracking and Surveillance*, Dec. 2009, pp. 1–6.

[139] R. Fisher, J. Santos-Victor, and J. Crowley, "CAVIAR: Context aware vision using image-based active recognition," 2005.

[140] M.A. Patricio et al., "Discrete optimization algorithms in real-time visual tracking," *Applied Artificial Intelligence*, vol. 23, pp. 805–827, 2009.

Appendix A

Objectness-based Tracker Drift Detection

A.1 Introduction and Related Work

Object detection can significantly support object tracking. Objects can be detected using saliency, sliding window, and objectness approaches. Saliency object detection [42–45] identifies the location of objects based on image information such as contrast. It has been widely used in different applications, including object recognition, image segmentation, and video summarization. In sliding window paradigm [35], a classifier is first trained to distinguish windows containing instances of a given class from all other windows, and then used to score every window in a test image. Local maxima of the score localize instances of the class. This approach detects specific class of objects, such as vehicles or humans, which is not appropriate for automated applications such as object tracking that track different types of objects. In addition, it is computationally intensive. Recently, objectness measures have become an active research area in object detection [38, 136]. Objectness measures attempt to generate a small set (few hundreds or thousands) of object regions that cover every object in the input image, regardless of the specific categories of those objects (generic over classes). Compared with traditional sliding window approach, estimating object proposals in a pre-processing stage has the following advantages: 1) better accords with our human visual system behavior which perceives objects before identifying them [38]; 2) speeds up the computation by reducing the search locations, especially when the number of object classes that need to be detected is high [37].

Objectness approaches [38, 136] are related to interest point detectors (IPS) and saliency models. IPS respond to local textured image neighborhoods and are widely used for finding image correspondences [40] and focus on individual points instead of the entire object(s) in image scene. Generally, the object proposals such as objectness approaches consider saliency as a useful cue for measuring objectness of a region [38]. Class-specific saliency models define, as a salient region, the visual characteristics that best distinguish a specific object class such as vehicle or human from others [41]. Class-generic saliency models [42–45] measure the saliency of pixels as the degree of uniqueness of their neighborhood relative to the surrounding region.

The objectness-based object detection approach in [38], the model is explicitly trained to distinguish objects with a well-defined boundary in space, from amorphous background elements, such as grass and road. The objectness measure [38] combines, in a Bayesian framework, several image cues measuring characteristics of objects, such as those appearing different from their surroundings and having a closed boundary. The object detection process in [38] is computationally intensive and shows inaccuracy when applied to low resolution images.

The estimated tracking output is a bounding box (BB) and in this appendix, we propose to use objectness measures to quantify how likely such BB contains an object (of any class) in order to detect if tracking drift tends to occur, and hence drift correction can be applied. Our method uses both the relative area and relative score of the detected candidate box among all objectness candidates to detect if the target inside the estimated BB is likely an object.

A.2 Proposed Method

Edge-based Objectness (EBP) [136] detects candidate object windows (or boxes) using edges which provides a sparse yet informative representation of an image. The number of contours that are wholly contained inside a box is used as an indication of the likelihood of the box to contain an object. EBP measures the number of edges that exist inside the box minus those that are members of contours that overlap the box boundary returning a ranked set of a few thousand top-scoring boxes. EBP in [136] is significantly more accurate while being twelve times faster to compute than the current state-of-the-art objectness measues [38]. We propose to use the maximum scored window in the estimated BB B_t to decide if the target started to drift, as shown in Figure A.1.

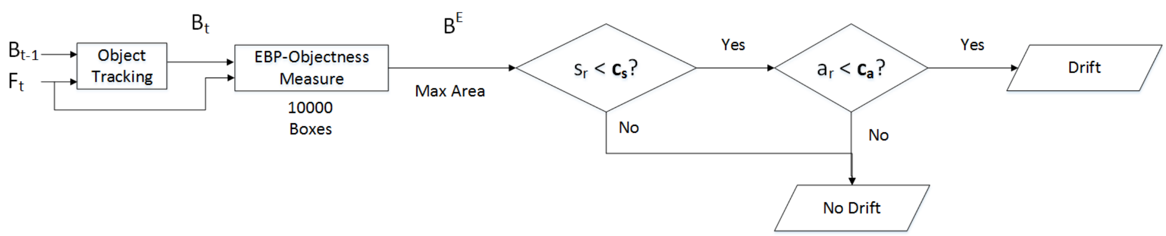


Figure A.1: Block diagram of the proposed edge-based objectness for drift detection.

The EBP candidate boxes have two pieces of information: area and score. The candidate box of maximum area and the candidate box of maximum score have shown to be complementary to each other (meaning if used together they will provide more information than using each separately) as shown in examples in Figure A.2. As can be seen, for different video sequences, the candidate box with maximum area and maximum score relative to other boxes at frame 40 shows two important regions of the target object inside the estimated tracking output box. Figure A.2 shows that the objectness measure can detect the box that includes the target object inside the estimated B_t without prior information. Hence, the two regions with maximum score and maximum area can be used to detect tracking drift, as we show next.

We select the box B^E among all candidate boxes, having largest area as it is more likely to include the target object (able to capture the whole object inside). We use the relative area a_r of B^E to the area of B_t to indicate the importance of the selected candidate box as

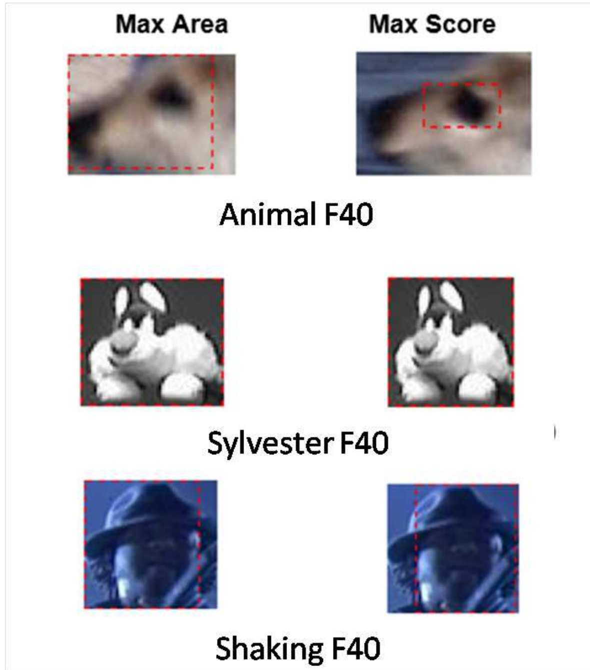


Figure A.2: Candidate box of maximum area and maximum score using edge-based objectness.

$$a_r = Area(B^E)/Area(B_t), \tag{A.1}$$

where $Area(B^E)$ is the area of the selected candidate box of maximum area inside B_t and $Area(B_t)$ is the area of the estimated B_t. Since the maximum score is not upper-bounded, we thus use the relative score s_r of B^E to the maximum score among all candidate boxes to indicate the importance of the selected candidate box as

$$s_r = score(B^E)/s_{max}, \tag{A.2}$$

where $score(B^E)$ is the score of the selected candidate box of maximum area inside B_t and s_{max} is the maximum score over all detected candidate boxes inside B_t. We correct drift, if both s_r and a_r are high according to some selected constants as

$$oDrift = \begin{cases} 1 & : (s_r \leq c_s) \wedge (a_r < c_a), \\ 0 & : otherwise \end{cases} \tag{A.3}$$

The constants c_s and c_a are selected as 0.8 and 0.5, respectively for scale invariant trackers (such as STRUCK [2] and KCF [10]) and 0.95 and 0.65 for scale variant trackers (such as SAMF [11], DSST [13], and STAPLE [3]). When the drift is detected, a drift correction can be applied (such as the proposed method in chapter 3)

A.3 Objective Results

Table A.1 shows the improvement in AOR, CLE, and FR, using the proposed objectness-based drift detection (and segmentation-based drift correction as in chapter 3, for five trackers relative to their corresponding original tracking quality over 11 different test sequences (Deer, FaceOcc2, Boy, Sylvester, David, Car4, Girl, Singer1, Shaking, Walking2, and Board) that cover different challenges [56] (better quality is shown in bold). Over all frames of all test video sequences, the proposed method gives better quality; no outliers were noted. As can be seen, all trackers have achieved better tracking quality compared with their corresponding original versions. KCF and STAPLE trackers showed the best improvement among all tested trackers.

Table A.1: Average improvement of the proposed objectness-based drift detection and correction method.

Trackers	Overlap Ratio (AOR)	Center Location Error (CLE)	Failure Rate (FR)
STRUCK	0.581	17.6	0.173
STRUCK-SegTrack-EBP	**0.594**	**14.73**	**0.103**
Improvement	2.23%	16.30%	40.46%
KCF	0.518	24.46	0.242
KCF-SegTrack-EBP	**0.596**	**12.31**	**0.057**
Improvement	15.05%	49.76%	76.44%
SAMF	0.64	13.43	0.097
SAMF-SegTrack-EBP	**0.691**	**9.38**	**0.053**
Improvement	7.96%	30.15%	45.36%
DSST	0.737	9.17	0.067
DSST-SegTrack-EBP	**0.745**	**7.97**	**0.062**
Improvement	1.08%	13.08%	7.46%
STAPLE	0.654	21.76	0.217
STAPLE-SegTrack-EBP	**0.731**	**8.35**	**0.044**
Improvement	11.77%	61.62%	79.72%

Figures A.3, A.4, and A.5 show the improvement in AOR, CLE, and FR, by the proposed method applied to the KCF tracker over frames of the above mentioned test sequences. We compare the achieved improvement with our proposed saliency-based drift detection and correction in chapter 3 (blue solid curve). As shown, the proposed objectness-based has achieved on average a slightly better tracking quality in AOR and CLE and slightly worse in FR.

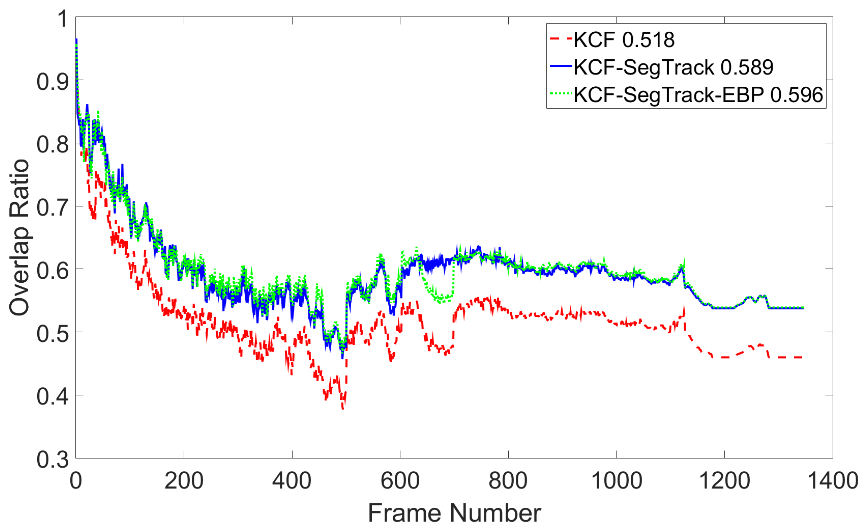


Figure A.3: AOR of base and modified KCF tracker using objectness.

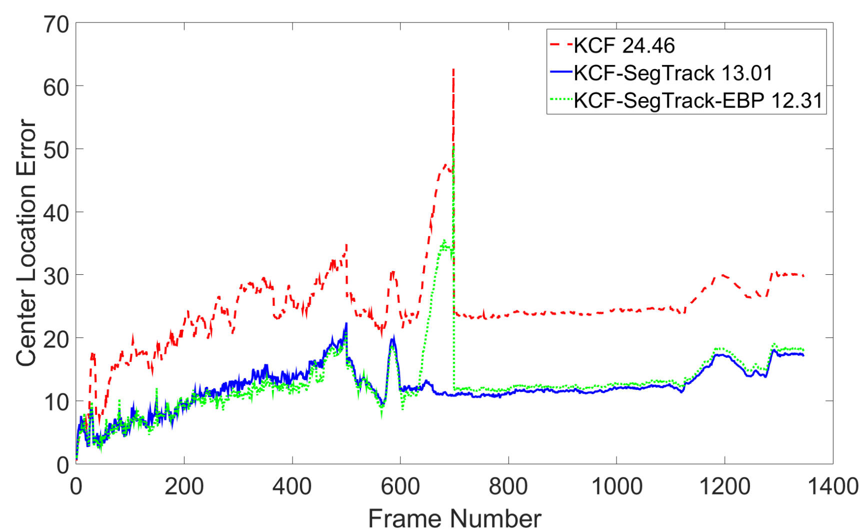


Figure A.4: CLE of base and modified KCF tracker using objectness.

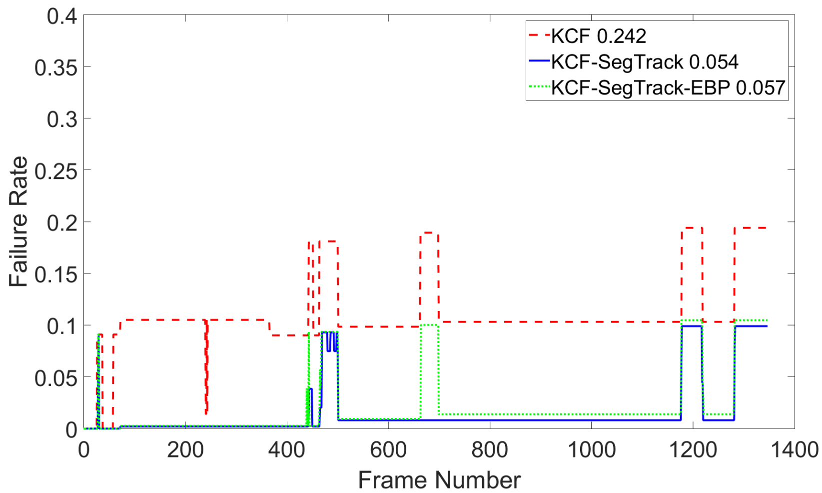


Figure A.5: FR of base and modified KCF tracker using objectness.

Figures A.6, A.7, and A.8 show the improvement in AOR, CLE, and FR, by the proposed method applied to the STAPLE tracker. As can be seen, the quality of our proposed method (green) outperforms that of the corresponding original trackers (dashed red) for AOR, CLE, and FR, while being similar to the results of saliency-based drift detection and correction proposed in chapter 3.

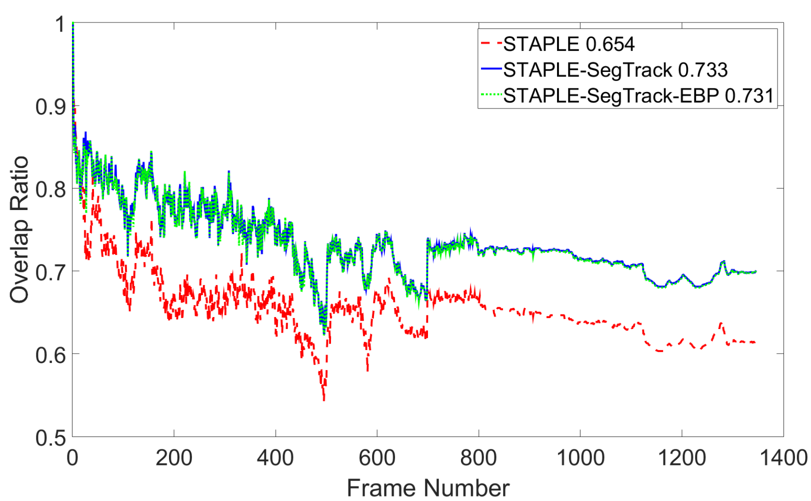


Figure A.6: AOR of base and modified STAPLE tracker using objectness.

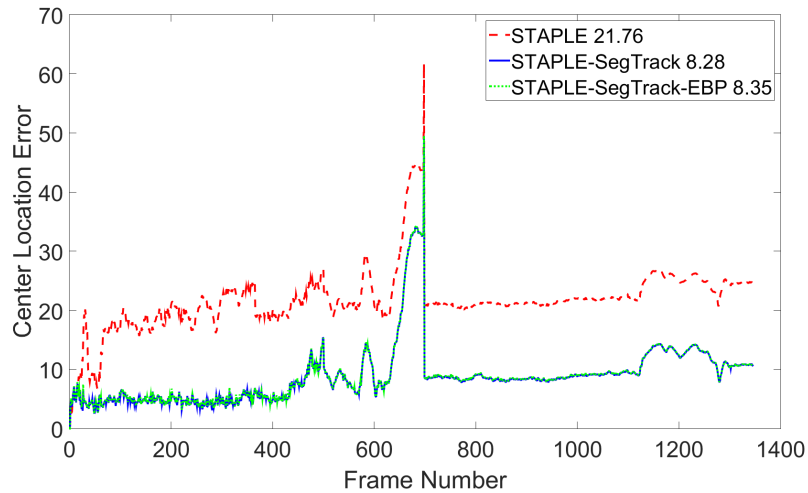


Figure A.7: CLE of base and modified STAPLE tracker using objectness.

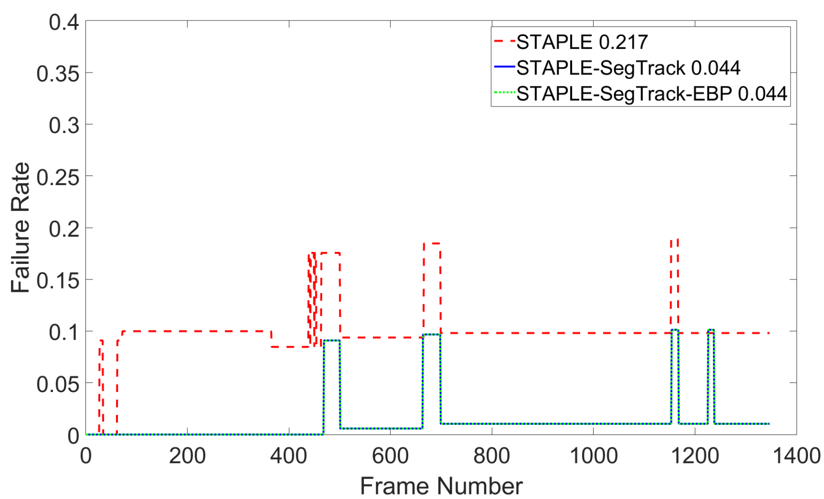


Figure A.8: FR of base and modified STAPLE tracker using objectness.

A.4 Conclusion

In conclusion, we can see that the use of objectness for drift detection gives slightly better results compared to using saliency features. The difference is that saliency features use the global contrast differences and spatial coherence of the given image while the objectness uses the number of contours extracted from edges which provides a sparse yet informative representation of an image. Both saliency and objectness are computationally efficient. The objectness measure and saliency features capture different pieces of information about the target object and the integration of both objectness and saliency together for drift detection is expected to achieve better results than the individual features.

A.5 Ground-Truth based Reference to Drift Detection

Tracking algorithms need to address possible drifts to take recovery measures in order to re-guide the tracker. To provide a reference for drift detection and correction methods, in this sectoin we propose to detect the tracking drifts based on the ground truth annotations, using two thresholds c_a and c_c based on the average overlap ratio (AOR) and the normalized center location error ($NCLE$), respectively [57]. At F_t, the $NCLE$ is the CLE in x and y directions relative to the dimensions of the ground truth BB. Figure A.9 illustrates the flowchart of the ground-truth based drift detection.

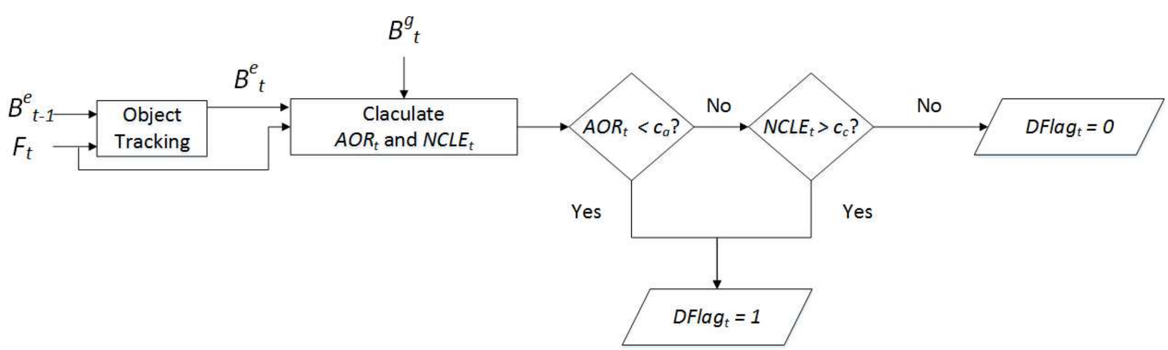


Figure A.9: Ground truth based drift detection.

As shown, given the annotated BB B_t^g and the estimated BB B_t^e at frame F_t, drift is detected as

$$GDFlag_t = \begin{cases} 1 & : (AOR_t < c_a) \vee (NCLE_t > c_c) \\ 0 & : otherwise, \end{cases} \tag{A.4}$$

where $AOR_t = AOR(B_t^e, B_t^g)$ and $NCLE_t = NCLE(B_t^e, B_t^g)$ at F_t. Once drift is detected, the flag $GDFlag_t$ is set to trigger the drift correction process. c_a and c_c are calculated as follows. For each tracker i, for video sequence j, the minimum overlap value AOR_{fj} over all video frames is calculated and then, the average of all minima of all videos is calculated as (A.5).

$$t^{min}_{AOR_i} = \frac{1}{J}\sum_{j=1}^{J}\min_{1\leq f\leq F}(AOR_{fj}) \tag{A.5}$$

where J is the number of video sequences, f is the frame number, and F is total number of frames. The process is repeated for T trackers and, the threshold c_a is calculated as the minimum average value over all trackers as (A.6).

$$c_a = \min_{1\leq i\leq T}(t^{min}_{AOR_i}) \tag{A.6}$$

c_c is set as follows. For each tracker, for each video sequence, the average $NCLE$ between B^e_t and B^g_t over all frames is calculated. The average of such normalized error over all sequences is then calculated for that tracker as (A.7).

$$t^{avg}_{NCLE_i} = \frac{1}{J}\sum_{j=1}^{J}(NCLE(v_j)), \tag{A.7}$$

where $NCLE(v_j)$ is the normalized error of video j according to tracker i. Then the average of the videos whose $NCLE$ is below the calculated average is found as (A.8).

$$t_{NCLE_i} = \frac{1}{J'}\sum_{j=1}^{J'}(NCLE(v_j))_{1\leq j\leq J' \wedge (NCLE(v_j)<T^{avg}_{NCLE_i})} \tag{A.8}$$

where J' is the number of videos of error below $T^{avg}_{NCLE_i}$. The process is repeated for all trackers. c_c is then selected as half of the minimum average value in all trackers as (A.9).

$$c_c = \frac{1}{2}.\min_{1\leq i\leq T}(T_{NCLE_i}) \tag{A.9}$$

For the KCF, SAMF, STRUCK, and DSST trackers, and the 11 test videos that cover all challenging attributes, it was found that $c_a = 0.220$ and $c_c = 0.066$. The drift correction is applied through segmentation as illustrated. The obtained results show that applying segmentation only when a drift is detected has led to both better tracking quality and performance than applying segmentation at each frame. Figures A.10, A.11. and A.12 show the AOR, CLE, and FR plots of the proposed ground truth-based drift detection and correction applied to KCF tracker (dashed red) vs. applying the segmentation-based drift correction at each frame (dotted blue). As illustrated, applying segmentation only when a drift is detected shows better tracking quality over all sequences as well as better tracking performance.

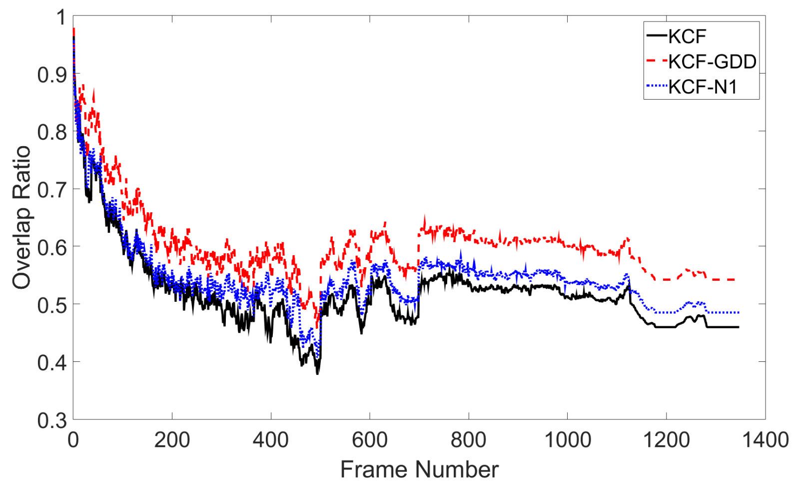


Figure A.10: AOR plot of ground-truth based drift detection (KCF-GDD) versus drift correction at each frame (KCF-N1) for KCF.

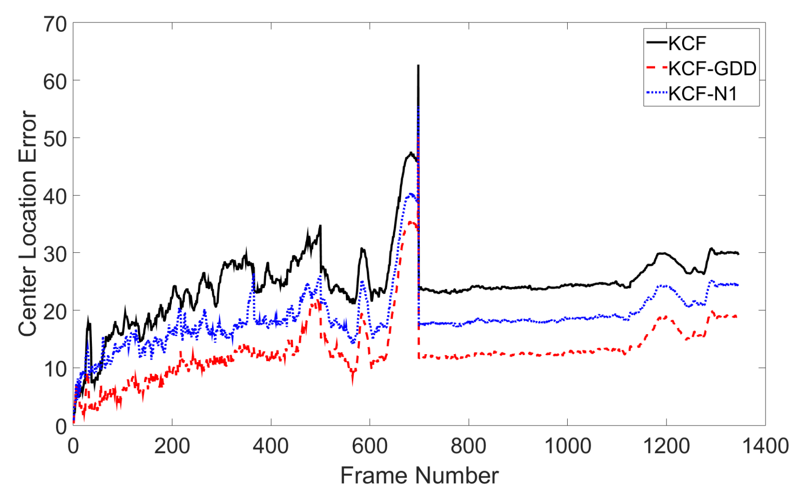


Figure A.11: CLE plot of ground-truth based drift detection (KCF-GDD) versus drift correction at each frame (KCF-N1) for KCF.

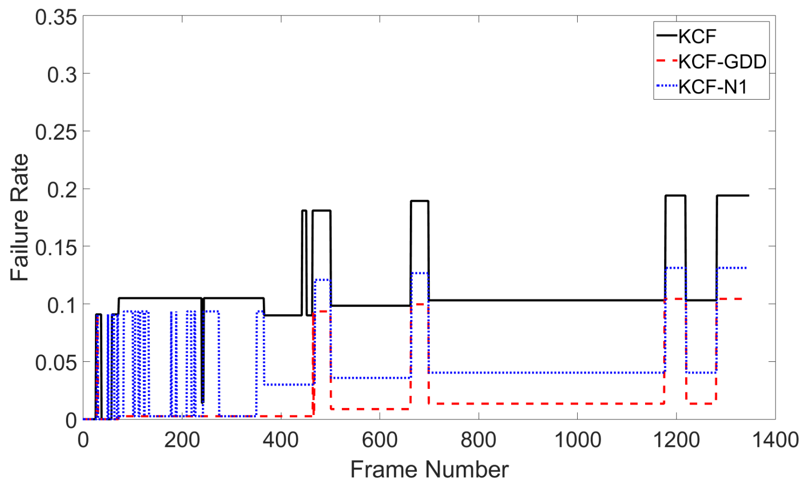


Figure A.12: FR plot of ground-truth based drift detection (KCF-GDD) versus drift correction at each frame (KCF-N1) for KCF.

Appendix B

Framework for Parameter Weighting and Selection for Object Tracking Algorithms

B.1 Abstract

A little research attention has been given to study tracking algorithms' sensitivity to the tracking parameters, and to select the best performing parameter sets (or configurations). Long and tedious parameter fine tuning process is usually adopted to choose the best configuration that has a strong influence on the tracking quality, particularly with the diverse nature of video signals. This chapter proposes a framework for selecting the best configurations of a tracking algorithm using three quality metrics (failure rate, average overlap ratio, and normalized center location error) coupled with the median absolute deviation. Our framework weights the parameters of the tracking algorithm according to their sensitivity to each of these parameters with respect to performance measures and uses a scoring mechanism to suggest the best and second best configurations. We tested our framework on different trackers and found that it either suggests a new configurations that outperforms the default one or confirms the tracker default configuration.

B.2 List of Symbols

Symbol	Description
N_β	Number of parameters
$[y_{kmin}, y_{kmax}]$	Range of parameter k
N	Number of choosen configurations
N_v	Number of test sequences
N_p	Number of metrics
w_k	Sensitivity weight of parameter k
y_k^Δ	Increment step of parameter k
N	Number of trackers
$\{s_{ij}\}$	Score vector for all scored trackers
c_i	Configuraion number i
d_w	Average number of variations of all parameters
d_q	Median absolute deviation threshold

B.3 Introduction

Video object tracking has remained a challenging task due to its diverse applications, scene conditions, and parameters that often require manual fine tuning for acceptable tracking results. A variety of trackers have been developed to address these challenges [2, 8–13]. Tracking algorithms are usually more sensitive to certain parameters than others. The selection of the best tracking parameter configuration (set of parameter values) is important for fair comparison of tracking algorithms. However, little research attention has been given to select the best configuration. The performance evaluation methodologies, such as Online Tracking Benchmark OTB [57], Visual Object Tracking challenge VOT [91–93, 96], and NUSPRO [94], use the default configurations (given by the respective authors) for evaluating object tracking algorithms. A tedious effort is usually spent by authors to fine tune the parameters during the experimental phase to achieve the best results. Manual fine tuning [137] does not however guarantee optimal results.

The rest of this Appendix is organized as follows. Section B.4 presents related work and its differentiation to the proposed approach, which is introduced in Section B.5. Section B.6 discusses the obtained results. Section B.7 concludes the work.

B.4 Prior Work

Few research papers have addressed parameter selection, scoring, and weighting for object tracking algorithms. PETS [138] and CAVIAR [139] workshops are among the earliest ones to put efforts on comparing different trackers. They provided public datasets with different aspects of tracking scenarios to facilitate testing of different trackers, so that certain evaluation metrics can be used to compare the results. VOT2013 [91] has introduced a ranking-based methodology that accounted for statistical significance of the results. VOT2013 used accuracy and robustness as two weakly-correlated performance measures for the trackers' evaluation. However, the default parameter configurations were used for evaluation, or, when not available, were set to "reasonable" values without rigorous testing. VOT2014 [92] and VOT2015 [93] followed the same procedure as VOT2013 to select the parameters' configuration for all trackers. In [94], Y. Wu et al. used the overlap ratio for evaluation and ranking of the tracking performance in which the trackers were set to their default parameters.

In [115], Phu et al. proposed an approach that learns how to tune the tracker parameters online to cope with the tracking context variations. However, it needs an off-line training step in order to learn such variations from different sequences. Moreover, the effect of different parameters on the tracking quality (parameter sensitivity) is not measured. In [137], Adenso and Laguna presented a support tool (CALIBRA) for fine-tuning algorithms based on fractional factorial experimental design coupled with local search procedure to facilitate the task of finding parameter values for algorithms. Specifically, CALIBRA attempted to find the best values for up to five parameters based on local search procedure. However, the significance of interactions among parameters is not exploited. The difference between [137] and our framework is that a) ours supports any number of parameters, and b) ours excludes any configuration that has minor effect on the tracking performance, and hence can reflect the importance of interactions among parameters. In [140], Patricio et al. proposed a formulation of the data association problem in visual tracking systems as a discrete optimization where for every frame, the detected blobs are assigned to active tracks in such way to maximize the tracking accuracy. They used different search algorithms to find the best tracking configuration to match hypothesis over time during tracking. However, they mentioned that they are willing to develop a self-tuning version of the local search algorithms to choose the right parameters for every instance of a problem.

Our work differs from those in the literature. It searches the best parameter configuration of a tracking algorithm without human intervention. The best configuration is searched based on the quality values according to different evaluation metrics. The selected configurations are scored to find the one with the highest possible tracking quality with respect to performance measures. The proposed framework then assigns a weight to each parameter based on its influence on the performance of the tracking algorithm. In addition, it has the flexibility to control the search range and increments of all parameters according to the reqired accuracy of the selected configurations.

B.5 Proposed Framework

The proposed framework comprises three steps as shown in Figure B.1: data entry, parameter weighting, configuration scoring. The data entry step reads the following: the tracker parameters $\{\beta_k; k = 1, \cdots, N_\beta\}$, N_β is the number of parameters; the range of each parameter $y_k = [y_{k_{min}}, y_{k_{max}}]$; parameters' variation (increment) step y_k^Δ; the ground truth bounding boxes (BBs); the performance evaluation metrics $p_j, j = 1, \cdots N_p$, N_p is the number of metrics used to calculate the quality values q_{ijl} for each configuration $\{c_i; i = 1, \cdots, N\}$, N is the number of chosen configurations from the pool of all possible configurations according to metric p_j over a test sequence $\{v_l; l = 1, \cdots, N_v\}$, N_v is the number of test sequences; and the dispersion estimator measure. Then the parameters' weighting step calculates a sensitivity weight vector $\{w_k\}$ for all parameters. For each p_j, for each v_l, the scoring step applies the dispersion measure on the vector of quality values $\{q_{ijl}\}$ for the N configurations and assigns score vector $\{s_{ij}\}$ for each configuration c_i according to metric p_j as the sum of scores over all sequences, where $1 \leq s_{ij} \leq N_v$. The proposed framework uses three known evaluation measures: the average overlap ratio

(AOR), the normalized center location error ($NCLE$), and the failure rate (FR).

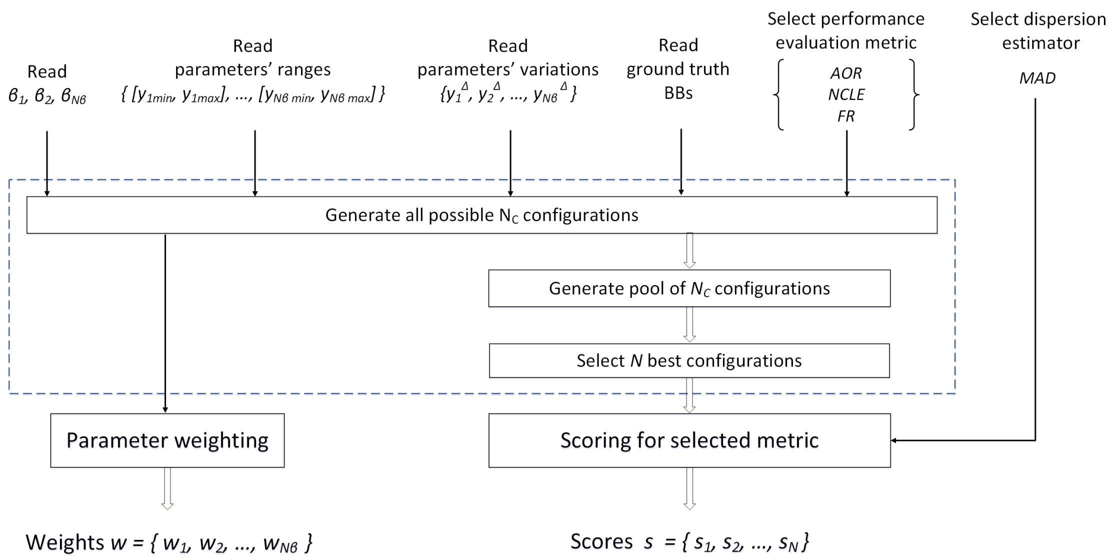


Figure B.1: Block diagram of the proposed parameter selection and weighting framework.

B.5.1 Parameter Weighting

Tracking algorithms usually incorporate parameters that need to be tuned to get better results. Some parameters may have slight or no influence on a tracker's quality. We propose a parameter weighting mechanism which tests all parameters of the tracking algorithm separately and evaluates their sensitivity weights. Given a certain tracker with set of parameters $\{\beta_k; k = 1, \cdots, N_\beta\}$, each parameter with a range $y_k = [y_{k_{min}}, y_{k_{max}}]$ and increment step y_k^Δ, our weighting mechanism first creates a set of all $N_k = \frac{y_{k_{max}} - y_{k_{min}}}{y_k^\Delta}$ possible configurations $\{c_m; m = 1, \cdots, N_k\}$ with respect to each parameter β_k by varying its range y_k by an increment step of y_k^Δ, while keeping all other parameters constant to a value from their respective range and increment. Thus the total number of all possible configurations of all parameters is $\prod_{k=1}^{N_\beta} N_k$. Then, our weighting mechanism scores each configuration c_m according to performance measure j for parameter k as

$$t_{mjk} = \begin{cases} 1, & \max\{Q_{mjk}\} - \min\{Q_{mjk}\} \geq d_w, \quad \forall m \\ 0, & \text{otherwise,} \end{cases} \tag{B.1}$$

where t_{mjk} is the score of configuration c_m according to performance measure j for parameter k, $\{Q_{mjk}\}$ is the set of individual quality values Q_{mjk} of configuration m according to performance measure j for parameter k, and the threshold d_w is defined as

$$d_w = \frac{|\max\{Q_{mjk}\} - \min\{Q_{mjk}\}|}{a}, \quad \forall m, k \tag{B.2}$$

where $a = \frac{\sum_{k=1}^{N_\beta} N_k}{N_\beta}$ is the average number of variations of all parameters. Next, the weight w_k of parameter β_k is calculated as the average of scores s_{ijk} assigned to each configuration i according to metric j as

$$w_k = \frac{\sum_{k=1}^{N_k} s_{ijk}}{N_k}. \tag{B.3}$$

The process is repeated over all set of parameters $\{\beta_k\}$. Finally, a weight vector $w = \left\{w_1, w_2, \ldots, w_{N_\beta}\right\}$ is calculated. Then, the framework outputs normalized weights w_k as $\bar{w}_k = \frac{w_k}{\sum_{k=1}^{N_\beta} w_k}$ to obtain the final sensitivity weight for each parameter.

B.5.2 Configuration Selection and Scoring

The variation of all parameters, with their ranges and increments, creates a large number of configurations. We thus need to filter out less performing ones. We select the N configurations among all possible configurations that either maximize AOR or minimize $NCLE$ or FR. We pick any configuration which yields a quality value within a threshold from the quality value of the best configuration, which is the one that gives the highest (or lowest) quality value among all configurations according to selected metric p_j.

Due to the possible presence of outliers in numerical data of configurations' measures, counting the number of best and 2^{nd} best scores is not reliable to measure their performance, because it neglects the deviation that may exist in the data. For each test sequence, we therefore define a deviation threshold d_q based on a dispersion measure of the vector of quality values $\{q_{ijl}\}$ of N configurations which evaluates a set's close affiliation to either the best or the second best score. We use the median absolute deviation (MAD) dispersion measure of the quality vector $\{q_{ijl}\}$ for the N configurations using performance measure j on the sequence l as

$$d_q = Median|q_{ijl} - Median\{q_{ijl}\}|, i = 1, \cdots, N, \tag{B.4}$$

where q_{ijl} is the individual quality value for the configuration i according to performance measure j for sequence l. We process sequences separately and for each sequence, we score all configurations and then count scores over all sequences for each quality metric. The process of the scoring mechanism for the N configurations over all sequences is summarized in Algorithm B.1.

Algorithm B.1: Scoring of N configurations for a metric p_j.

Data: Quality values $\{q_{ijl}\}$; $\{i = 1, \cdots, N, j = 1, \cdots, N_p\}$ of all $\{v_l\}$
Result: Scores $\{s_{ij}\} = \{s_{1j}, \cdots, s_{Nj}\}$ for all N configurations with respect to each p_j

for *a performance measure* p_j **do**
  for *a tracker* i **do**
    $s_{ij} = 0$;
  for *each test sequence* v_l **do**
    $d_q = MAD\{q_{ijl}\}$;
    $O = Best\{q_{ijl}\}$;
    for *each unscored configuration* c_i **do**
      if $|q_{ijl} - O| < d_q$
        score = 1;
      else
        score = 0;
      end;
      $s_{ij} = s_{ij}$ + score;

$Best\{q_{ijl}\}$ is the best value among the set of quality values of all N configurations for the test sequence v_l (either the maximum value for AOR or the minimum for both $NCLE$ and FR); $\{s_{ij}\}$ is a score vector that counts the scores of each configuration i over all test sequences according to metric j. Scores for each configuration are aggregated over all sequences to find the scores vector $\{s_{ij}\}$ for each metric j. Then, the configuration(s) with the maximum count is (are) selected as the best configuration(s). The ones that are not scored as best, contribute to another round in order to choose the second best configuration(s) using the same Algorithm B.1 with a new threshold d_q selected based on the remaining configurations' quality values. This is repeated till all N configurations are scored.

B.6 Results and Analysis

B.6.1 Experimental Setup

We run each simulation five times on each test sequence to obtain fair statistics on the mentioned performance measures. We then determine the results for every configuration over all sequences. We have tested our framework with three well-known trackers, STRUCK [2], KCF [10], and STAPLE [3] and three performance measures on 50 test sequences [56]. Struck [2] is a framework for adaptive visual object tracking based on structured output prediction. The framework uses a kernelized structured output support vector machine (SVM), which is learned on-line to provide adaptive tracking. STAPLE tracker [3] combines two image patch representations to learn a model that is inherently robust to both color changes and deformations. Two independent ridge regression problems are solved, exploiting the inherent structure of each representation to maintain real-time performance. STAPLE combines the scores of template and histogram models in a dense translation search, that are learned independently, enabling greater accuracy. For the KCF [10] tracker, it performs training and detection to discriminate an object's appearance from its environment. Henriques et al. in [10], used analytical tools of circulant matrices to reduce data storage and increasing the tracking speed.

STRUCK parameters are search radius (SR), budget size (BS), and regularization (RP), which are set by default to 30, 100, and 100, respectively. The authors in [2] stated that the RP parameter of STRUCK Tracker has little influence on tracking quality and thus is not included in our simulations. STAPLE parameters are template learning rate (TLR), histogram learning rate (HLR), histogram bins (NBIN), merge factor (MF), fixed area (FA), and hog cell size (HCS), which are set to 0.01, 0.04, $32 \times 32 \times 32$, 0.3, 150×150, and 4×4. KCF parameters are feature bandwidth (FB), adaptation rate (AR), spatial bandwidth (SB), and regularization (RP), which are by default set to 0.5, 0.02, 0.1, and 0.0001. Table B.1 gives the parameters' ranges and increments used in our simulations for STRUCK, KCF, and STAPLE. HCS and NBIN parameters use multiplicative increments.

Table B.1: Ranges and increments of the parameters of STRUCK, KCF, and STAPLE trackers.

Tracker	Parameters	Minimum	Maximum	Increment
STRUCK	SR	20	40	2
	BS	20	100	20
Staple	TLR	0.005	0.02	0.005
	HLR	0.03	0.005	0.05
	NBIN	16	64	2
	MF	0.2	0.4	0.05
	FA	125	175	25
	HCS	2	8	2
KCF	FB	0.5	1.5	0.2
	AR	0.02	0.08	0.02
	SB	0.1	0.3	0.1
	RP	0.0001	0.0005	0.0002

B.6.2 Parameter Weighting

Our parameter weighting mechanism assigned a normalized sensitivity weight to each parameter and separated non-sensitive parameters from their sensitive counterparts. We applied the weighting in (B.3) to measure the parameters' sensitivity of STRUCK, KCF, and STAPLE trackers. Weights of these trackers' parameters are listed in Table B.2 on a per-metric basis. As can be seen, for STRUCK, SR has more weight than BS across all metrics. For STAPLE, results showed that our weighting mechanism indicates that the HLR parameter has no influence, while the other parameters have different weights for each quality metric view point. For AOR, STAPLE depends entirely on

TLR, FA, and HCS parameters. For NCLE, STAPLE also depends on TLR, HCS, and MF that assigned higher weight than FA. For FR, STAPLE shows to depend mainly on both TLR and NBIN parameters. For KCF, the obtained results show that AR and SB parameters are the most effective, with SB having the highest weight (most influential) across all three metrics. Parameter weighting is important for improvement of the tracking algorithms as it focuses on parameters to which the algorithm is more sensitive.

Table B.2: Generated parameters' weights w_k.

Tracker	Parameters	Generated parameter weights		
		AOR	NCLE	FR
STRUCK	SR	0.5732	0.5375	0.5412
	BS	0.4268	0.4625	0.4588
Staple	TLR	0.2443	0.2713	0.2698
	HLR	0	0	0
	NBIN	0.0992	0.1938	0.2407
	MF	0.229	0.2171	0.1852
	FA	0.0458	0.1008	0.127
	HCS	0.3817	0.2171	0.1772
KCF	FB	0	0	0
	AR	0.2653	0.4734	0.2299
	SB	0.7347	0.5266	0.7701
	RP	0	0	0

B.6.3 Configuration Scoring

Table B.3 shows the scoring results of the N best configurations for STRUCK tracker. The first column (set 30/100) in the table indicates the default set as given by respective authors. As can be seen, all configurations are scored according to their effect on the quality of each performance measure. In order to facilitate interpretation of scoring using the three separate performance measures $NCLE$, AOR, and FR, we combined their scoring as follows. For each configuration c_i, the median M_1 of best scores $\{s_{ij}\}$ across the $NCLE$, AOR, and FR is calculated as $M_1 = Median\{s_{ij}\}$, where $\{s_{ij}\}$ is a set of best scores according to metric j. Such median is useful as an estimator of central tendency and is robust against outliers. Similarly, the median of 2^{nd} best scores M_2 is calculated for each configuration c_i. As can be seen in Table B.3, among the N selected configurations, the two best scored configurations from tracking quality view point ($c1$ and $c2$) are ranked (bold) and they outperform the default STRUCK configuration according to $NCLE$, AOR, and FR. For example, the 40/80 configuration with best score has higher tracking quality according to M_1 and M_2. We also calculated a combined relative rank ($Rank$) of configurations c_i in Table B.3 as $Rank = 0.85M_1 + 0.15M_2$, which weights best scores higher than second best scores. As shown in Table B.3, this ranking measure confirms the M_1 scoring for the selected best configurations. The two best sets ranked as top also have the highest ranks among all trackers.

Table B.3: Scoring of parameter sets of STRUCK.

	STRUCK:NCLE													
	default	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13
SR	30	38	40	26	34	32	28	36	36	32	32	36	28	26
BS	100	100	80	80	100	60	20	100	80	40	80	20	60	100
Mean	0.14	0.13	0.13	0.14	0.14	0.13	0.12	0.13	0.14	0.13	0.12	0.14	0.13	0.14
# Best scores	9	15	17	14	10	13	11	10	8	10	13	10	13	11
# 2nd best scores	8	8	11	9	9	10	11	10	11	10	5	11	10	13

	STRUCK:AOR													
SR	30	38	40	26	34	32	28	36	36	32	32	36	28	26
BS	100	100	80	80	100	60	20	100	80	40	80	20	60	100
Mean	0.37	0.38	0.38	0.36	0.37	0.37	0.37	0.37	0.37	0.36	0.37	0.35	0.36	0.36
# Best scores	9	15	15	15	9	11	14	9	7	10	12	10	13	13
# 2nd best scores	15	10	11	8	8	10	9	9	12	9	8	6	8	13

	STRUCK:FR													
SR	30	38	40	26	34	32	28	36	36	32	32	36	28	26
BS	100	100	80	80	100	60	20	100	80	40	80	20	60	100
Mean	0.33	0.31	0.3	0.31	0.32	0.32	0.29	0.33	0.33	0.31	0.3	0.33	0.31	0.31
# Best scores	23	26	23	25	22	24	25	19	20	25	28	17	25	23
# 2nd best scores	8	5	11	8	7	7	8	11	7	4	3	13	9	8

	STRUCK:MED													
SR	30	**38**	**40**	26	34	32	28	36	36	32	32	36	28	26
BS	100	**100**	**80**	80	100	60	20	100	80	40	80	20	60	100
M1	9	**15**	**17**	15	10	13	14	10	8	10	13	10	13	13
M2	8	**8**	**11**	8	8	10	9	10	11	9	5	11	9	13
Rank	0.18	**0.28**	**0.32**	0.28	0.19	0.25	0.27	0.2	0.17	0.2	0.24	0.2	0.25	0.26

Table B.4 shows the scoring results of the N best configurations for STAPLE tracker. The first set 0.04/0.01/0.3/150/32/4 in the table indicates the default set as given by authors. As shown, all configurations are scored according to their quality values from each performance measure view point. As shown in Table B.4, among the N selected configurations, the two best scored ones ($c4$ and $c9$) according to the tracking quality are ranked (bold) and they outperform the default STAPLE configuration according to $NCLE$, AOR, and FR metrics. For example, the 0.04/0.02/0.35/150/64/2 configuration with best score has higher tracking quality according to $Rank$ measure and has achieved an improvement of 9.94%, 15.53%, and 30.43% for $NCLE$, AOR, and FR respectively. In addition, the selected $2nd$ best configuration 0.04/0.02/0.35/175/64/2 shares the same values with parameters of the best configuration except for FA parameter and slightly differs in tracking quality than the best one which reflects the accuracy of the weighting mechanism that has assigned a small weight to such FA parameter as per $NCLE$, AOR, and FR metrics. Table B.4 also shows that most of other selected configurations outperform the default one differently with respect to each metric which shows that the default configuration does not guarantee the best performance according to the tracking quality.

Table B.4: Scoring of parameter sets of STAPLE.

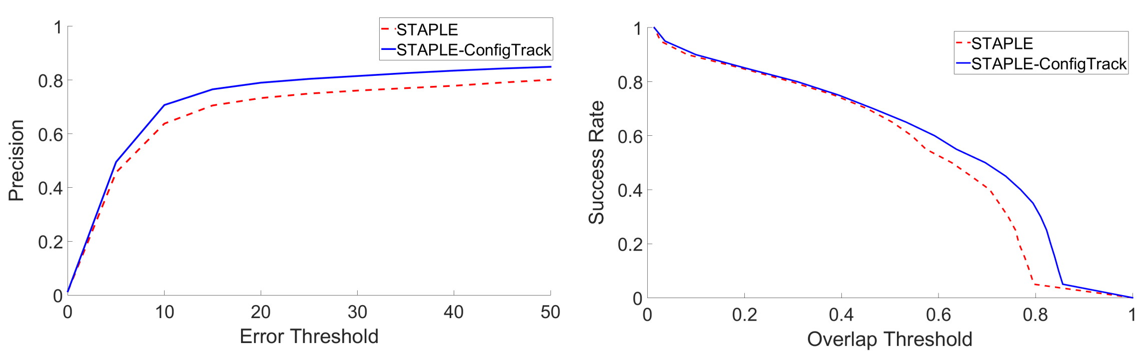
	Staple:NCLE											
	default	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11
HLR	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
TLR	0.01	0.01	0.01	0.01	0.02	0.01	0.01	0.02	0.02	0.02	0.02	0.02
MF	0.3	0.2	0.3	0.3	0.35	0.2	0.25	0.3	0.35	0.35	0.25	0.2
FA	150	125	175	150	150	175	150	175	150	175	150	175
NBIN	32	32	32	32	64	64	16	32	32	64	16	64
HCS	4	4	2	4	2	8	8	2	2	2	2	2
Mean	0.1	0.08	0.09	0.09	0.09	0.11	0.08	0.09	0.09	0.09	0.1	0.1
# Best scores	8	15	16	15	19	12	14	16	15	17	13	12
# 2nd best scores	13	7	16	8	12	10	8	9	13	16	12	15

	Staple:AOR											
HLR	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
TLR	0.01	0.01	0.01	0.01	0.02	0.01	0.01	0.02	0.02	0.02	0.02	0.02
MF	0.3	0.2	0.3	0.3	0.35	0.2	0.25	0.3	0.35	0.35	0.25	0.2
FA	150	125	175	150	150	175	150	175	150	175	150	175
NBIN	32	32	32	32	64	64	16	32	32	64	16	64
HCS	4	4	2	4	2	8	8	2	2	2	2	2
Mean	0.51	0.55	0.54	0.54	0.56	0.55	0.54	0.56	0.55	0.56	0.53	0.53
# Best scores	9	11	16	11	15	9	8	14	14	18	13	11
# 2nd best scores	11	11	11	11	8	6	9	13	7	12	5	9

	Staple:FR											
HLR	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
TLR	0.01	0.01	0.01	0.01	0.02	0.01	0.01	0.02	0.02	0.02	0.02	0.02
MF	0.3	0.2	0.3	0.3	0.35	0.2	0.25	0.3	0.35	0.35	0.25	0.2
FA	150	125	175	150	150	175	150	175	150	175	150	175
NBIN	32	32	32	32	64	64	16	32	32	64	16	64
HCS	4	4	2	4	2	8	8	2	2	2	2	2
Mean	0.23	0.18	0.2	0.2	0.16	0.16	0.17	0.17	0.19	0.17	0.21	0.2
# Best scores	25	28	30	31	36	30	35	31	31	34	29	30
# 2nd best scores	10	8	6	2	6	8	6	5	4	4	5	5

	Staple:MED											
HLR	0.04	0.04	0.04	0.04	**0.04**	0.04	0.04	0.04	0.04	**0.04**	0.04	0.04
TLR	0.01	0.01	0.01	0.01	**0.02**	0.01	0.01	0.02	0.02	**0.02**	0.02	0.02
MF	0.3	0.2	0.3	0.3	**0.35**	0.2	0.25	0.3	0.35	**0.35**	0.25	0.2
FA	150	125	175	150	**150**	175	150	175	150	**175**	150	175
NBIN	32	32	32	32	**64**	64	16	32	32	**64**	16	64
HCS	4	4	2	4	**2**	8	8	2	2	**2**	2	2
M1	9	15	16	15	**19**	12	14	16	15	**18**	13	12
M2	11	8	11	8	**8**	8	8	9	7	**12**	5	9
Rank	0.19	0.28	0.31	0.28	**0.35**	0.23	0.26	0.3	0.28	**0.34**	0.24	0.23

Concerning Precision and Success plots, Figures B.2. a) and B.2. b) show that the tracking quality of STAPLE tracker using the best performing configuration (blue) achieves better quality than using the default configuration (dashed red) with respect to different thresholds.



(a) Precision plot. (b) Success plot.

Figure B.2: Precision and Success plots of default and proposed selected configuration of STAPLE tracker.

Figures B.3 and B.4 show the *AOR* and *FR* plots for the best performing configuration compared with the default one over all test sequences. As shown, the selected configuration has better tracking quality along all test sequences.

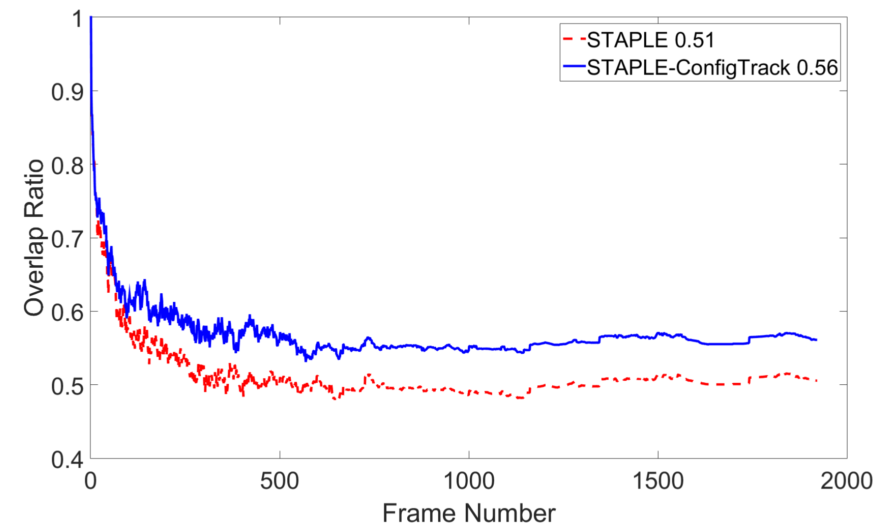


Figure B.3: AOR of default and best performing configuration of STAPLE tracker.

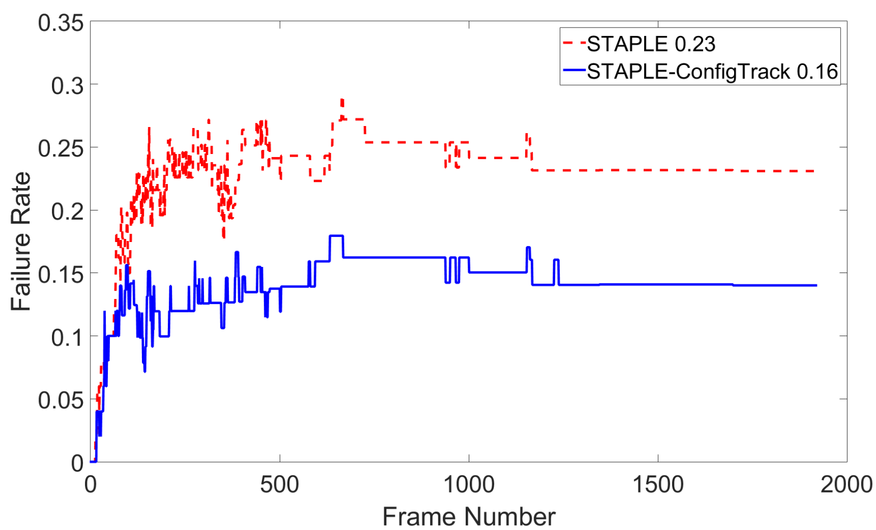


Figure B.4: FR of default and best performing configuration of STAPLE tracker.

Table B.5 shows the scoring results of configurations for KCF tracker. The set (0.5/0.02/0.1/0.0001) is the default as given by respective authors. Our framework confirms this set as the best set with the highest rank and best M_1 quality. Results show that the best scores measure is valuable and can reflect the actual tracking quality than the second best scores from all individual measure view points. As shown in Table B.5, the default configuration (bold) assigned the highest best score (best tracking quality) among all selected configurations.

Table B.5: Scoring of parameter sets of KCF.

	KCF:NCLE									
	default	c1	c2	c3	c4	c5	c6	c7	c8	c9
FB	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
AR	0.02	0.02	0.02	0.04	0.06	0.04	0.08	0.04	0.06	0.08
SB	0.1	0.2	0.3	0.1	0.1	0.2	0.1	0.3	0.3	0.2
RP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
Mean	0.12	0.16	0.19	0.16	0.18	0.19	0.19	0.2	0.21	0.21
# Best scores	32	22	11	20	16	12	15	10	10	4
# 2nd best scores	9	15	15	9	10	20	7	10	9	13

	KCF:AOR									
FB	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
AR	0.02	0.02	0.02	0.04	0.06	0.04	0.08	0.04	0.06	0.08
SB	0.1	0.2	0.3	0.1	0.1	0.2	0.1	0.3	0.3	0.2
RP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
Mean	0.4	0.31	0.24	0.34	0.31	0.27	0.31	0.22	0.21	0.25
# Best scores	33	18	9	17	16	11	12	7	5	8
# 2nd best scores	8	17	7	16	8	17	11	6	5	6

	KCF:FR									
FB	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
AR	0.02	0.02	0.02	0.04	0.06	0.04	0.08	0.04	0.06	0.08
SB	0.1	0.2	0.3	0.1	0.1	0.2	0.1	0.3	0.3	0.2
RP	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
Mean	0.27	0.38	0.47	0.34	0.36	0.43	0.39	0.48	0.48	0.46
# Best scores	32	20	15	25	23	20	20	12	12	13
# 2nd best scores	4	15	10	9	7	9	5	8	9	12

	KCF:MED									
FB	**0.5**	*0.5*	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
AR	**0.02**	*0.02*	0.02	0.04	0.06	0.04	0.08	0.04	0.06	0.08
SB	**0.1**	*0.2*	0.3	0.1	0.1	0.2	0.1	0.3	0.3	0.2
RP	**0.0001**	*0.0001*	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
M1	**32**	*20*	11	20	16	12	15	10	10	8
M2	**8**	*15*	10	9	8	17	7	8	9	12
Rank	**0.57**	0.39	0.22	0.37	0.3	0.26	0.28	0.19	0.2	0.17

From the above-mentioned discussion, it is clear that the proposed framework either confirms the default configuration of the tracking algorithm as in the case of KCF tracker, or selects a better configuration that significantly outperforms the default one as in STAPLE tracker.

B.7 Conclusion

This Appendix presents a framework is presented for selection and scoring of best performing set of control parameters of different tracking algorithms to achieve the highest tracking quality. The framework automatically weights all the parameters of selected tracker with respect to their influence on tracking quality from different performance measures view points. The proposed framework has been verified on three trackers, STRUCK, KCF, and STAPLE using three performance measures coupled with a MAD statistical dispersion metric. Results on 50 test sequences showed that the proposed scoring mechanism either suggests new configurations that outperform the default one or confirms the default configuration as being the best.

For future work, a multi-objective optimization approach that maximizes the AOR and minimizes the $NCLE$ and FR simultaneously is to be used to select the best configurations for a tracker.