Development of Efficient Object Tracking Algorithm using Machine Learning-based Meta-Learners

A Thesis

Submitted For The Degree Of

MASTER OF ENGINEERING

IN ELECTRONICS AND TELECOMMUNICATION ENGINEERING

by

Bibek Das

Roll No: 002210702005
Examination Roll No.: M4ETC24003

Under the guidance of

Prof. SHELI SINHA CHAUDHURI

Department of Electronics and Telecommunication Engineering

Jadavpur University

Kolkata 700032
West Bengal, India

May 2024

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains a literature survey and original work by the undersigned candidate, as part of his Master of Engineering in Electronics and Telecommunication in the Department of Electronics and Telecommunication Engineering. All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that I have thoroughly cited and referenced all material and findings which are not original to this research, as provided by these rules and conduct.

Name: Bibek Das
Examination Roll No.: M4ETC24003
Title: 'Development of Efficient Object Tracking Algorithm using Machine Learning-based Meta-Learners'

Signature of Candidate

FACULTY OF ENGINEERING AND TECHNOLOGY JADAVPUR UNIVERSITY

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled '**Development of Efficient Object Tracking Algorithm using Machine Learning-based Meta-Learners**' has been carried out by **Bibek Das**(Class Roll No: 002210702005, Examination Roll No.: M4ETC24003 and Registration No: 163531 of 2022-23) under my guidance and supervision and be accepted in partial fulfilment of the requirement for the degree of Master of Engineering in Electronics and Telecommunication in the Department of Electronics and Telecommunication Engineering.

Prof. Sheli Sinha Chaudhuri
Supervisor
Department of Electronics and
Telecommunication Engineering
Jadavpur University
Kolkata – 700032

Prof. Manotosh Biswas
Head of the Department
Department of Electronics and
Telecommunication Engineering
Jadavpur University
Kolkata – 700032

Prof. Dipak Laha
Dean
Faculty Council of Engineering
Jadavpur University
Kolkata – 700032

FACULTY OF ENGINEERING AND TECHNOLOGY JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL

The forgoing thesis titled '**Development of Efficient Object Tracking Algorithm using Machine Learning-based Meta-Learners**' is here approved as a creditworthy study of an engineering subject conducted and presented satisfactorily to warrant its acceptance as a precondition to the degree for which it was submitted. It is understood that the undersigned does not automatically support or accept any argument made, opinion expressed, or inference is drawn in it by this approval, but only approve the thesis for the reason for which it was submitted.

Committee on Final Examination for Evaluation of the Thesis

Signature of Additional External Examiner

Signature of Supervisor

Acknowledgements

I would like to extend my heartfelt gratitude to everyone who has supported and guided me throughout the journey of this thesis, '**Development of Efficient Object Tracking Algorithm using Machine Learning-based Meta-Learners**'.

First and foremost, I express my sincere appreciation to my thesis supervisor, **Dr. Sheli Sinha Chaudhuri**, Professor of the Department of Electronics and Telecommunication Engineering at Jadavpur University. Her invaluable guidance, insightful suggestions, and constant backing support have been essential for completing this thesis. Her positivity, confidence, and innovative ideas have been a constant source of motivation for me. I am truly fortunate to have had her as my mentor, guiding me like a guardian throughout this endeavor.

Furthermore, I am grateful to my lab seniors, for his suggestions, encouragement, and support, throughout all of which have been very helpful in helping me to improve my research work. I would also like to express my gratitude to my friends, fellow researchers, and the technical and non-technical Jadavpur University staff, whose help and support have been invaluable throughout this thesis endeavor.

I'd like to express my heartfelt gratitude to my parents and family, whose sacrifices have been the backbone of my journey. Without their unwavering support, love, and encouragement, I would not be where I am today. Their immeasurable contributions have shaped me into the person I am today, for which I am eternally grateful.

Bibek Das
Jadavpur University
Kolkata-32, West Bengal

Abstract

Object tracking is one of the most complex tasks in the field of computer vision. Although there are many deep learning-based algorithms designed to tackle issues such as occlusion, low visibility, and scale changes, they typically require large amounts of data, powerful processing capabilities, and extensive training time. On the other hand, conventional algorithms like KCF, TLD, and BOOSTING might not be effective at tracking objects consistently.

In order to bridge the existing gap, the primary focus of this study is to minimize the computing resources, training data, and training times required for object tracking. To accomplish this, a novel machine learning-based one-shot meta-learning approach has been proposed. The model combines traditional tracking algorithms, leveraging their predictions as inputs that are then refined through the machine learning model to improve overall performance.

The evaluation of the proposed model on the VOT2015 and OTB50 datasets confirms its exceptional performance compared to existing deep learning-based approaches, offering a promising solution for the intricate challenges of object tracking in computer vision. By effectively combining traditional methods with machine learning techniques, this approach not only showcases impressive effectiveness but also underscores its potential to provide robust and resource-efficient object tracking solutions for real-world applications.

Keywords: Computer Vision, Object Tracking, Machine Learning

Contents

1 Introduction **1**

2 Literature Review **3**
2.1 Point Tracking . . . 3
2.2 Kernel Tracking . . . 5
2.3 Silhouette Tracking . . . 6
2.4 Generative method . . . 7
2.5 Discriminative method . . . 7
2.5.1 Correlation model . . . 8
2.5.2 Deep learning model . . . 10
2.6 Different Motion Models in Object Tracking . . . 12
2.6.1 Linear motion model . . . 13
2.6.2 Nonlinear motion model . . . 13
2.7 Different selection based trackers . . . 13
2.7.1 Dictionary-based trackers . . . 13
2.7.2 Ensemble-based trackers . . . 13
2.7.3 Segmentation-based trackers . . . 14
2.8 Offline and Online Object Tracking . . . 14
2.8.1 Offline object trackin . . . 14
2.8.2 Online object tracking . . . 15

3 THESIS MOTIVATION AND CONTRIBUTION **16**

4 Proposed Methodology **17**
4.1 Experimental Setup: . . . 19
4.2 Data-set Description: . . . 19
4.3 Training Process : . . . 20

5 Result Analysis **22**
5.1 Qualitative Comparison : . . . 22
5.1.1 Success Plot : . . . 24
5.1.2 Center Location Error Plot : . . . 27
5.1.3 Precision Plot : . . . 28
5.1.4 Tracking results : . . . 29
5.2 Quantitative Comparison : . . . 40

5.2.1 State-of-the-art comparison : 40
5.2.2 Object Tracking Error : 41
5.2.3 Precision : 42
5.2.4 Expected Average Overlap (EAO) : 42
5.2.5 Success Rate : 46
5.3 Ablation Study 48
5.3.1 A study on different combinations of the base tracker 48
5.3.2 A study on different meta-learners 49
5.3.3 A study on the effect of feedback in the learning process 50

6 Conclusion And Future Works **51**

List of Figures

2.1 Basic flowchart of the visual object tracker 5
2.2 Taxonomy of the tracking methods 6
2.3 Flow chart of Generative models 7
2.4 Flow chart of Discriminative models 8
2.5 A general framework for correlation model 10
2.6 Classification Of Deep learning-based Object tracking 10
2.7 Siamese architecture based Deep learning model 11

4.1 Flowchart of the one-shot meta-learning system: The tracker stores previous tasks, guides meta-learning for regression model updates, and enables learning from the predicted output. 18
4.2 Illustrates the process of proposed tracking on some videos taken from Visual Object tracking benchmark dataset, VOT2015 [27]. 21

5.1 An illustration of Success Plot 24
5.2 A success rate comparison at different IOU thresholds of the proposed tracker and different existing tracker in Visual Object tracking benchmark dataset (VOT2015) [27]. The different color indicates the different tracker as shown above 25
5.3 Comparison of different tracker using Success plot in OTB 50 Dataset [47]. . 26
5.4 Comparison of different tracker using Success plot in OTB 50 Dataset [47]. . 27
5.5 Comparison of different tracker using Success plot in OTB 50 Dataset [47]. . 28
5.6 A center location error (Euclidean distance) comparison of the proposed tracker and different existing tracker in Visual Object tracking benchmark dataset (VOT2015) [27]. The different color indicates the different tracker as shown above 29
5.7 Comparison of different tracker using Center location error in OTB 50 Dataset [47]. 30
5.8 Comparison of different tracker using Center location error in OTB 50 Dataset [47]. 31
5.9 Comparison of different tracker using Center location error in OTB 50 Dataset [47]. 32
5.10 A precision at different center location threshold comparison of the proposed tracker and different existing tracker in Visual Object tracking benchmark dataset (VOT2015) [27]. The different color indicates the different tracker as shown above 33

5.11 Comparison of different tracker using Precision Plot in OTB 50 Dataset [47] 34
5.12 Comparison of different tracker using Precision Plot in OTB 50 Dataset [47] 35
5.13 Comparison of different tracker using Precision Plot in OTB 50 Dataset [47] 36
5.14 Tracking result in various challenging situations 38
5.15 Tracking result in various challenging situations 39
5.16 An illustration of Precision calculation 43
5.17 An illustration of Success Rate calculation 46

List of Tables

2.1 Different challenging situation 4
2.2 Different Categories of Tracking and Their Examples 7
2.3 Different features, Features Descriptor and their Advantage, Disadvantage and Example 9
2.4 Example of real-time and non-real-time tracking and their implementation method 15

4.1 RMSE Value and Processing Time of different individual Based Tracker . . 20

5.1 Different attribute description 22
5.2 Object properties used in this thesis 23
5.3 Success rate, Precision, Object Tracking Error, Expected Average Overlap and Frame Rate comparison of proposed tracker in Visual Object Tracking Benchmark dataset (VOT2015) [27] 40
5.4 Object tracking error comparison different tracking algorithms 41
5.5 Object tracking error comparison different tracking algorithms 42
5.6 Precision comparison of different tracking algorithms at threshold = 20 . . . 44
5.7 Precision comparison of different tracking algorithms at threshold = 20 . . . 44
5.8 Expected Average Overlap comparison of different tracking algorithms 45
5.9 Expected Average Overlap comparison of different tracking algorithms 45
5.10 Success Rate comparison of different tracking algorithms at IOU = 0.5 . . . 47
5.11 Success Rate comparison of different tracking algorithms at IOU = 0.5 . . . 47
5.12 RMSE and Frame Rate of the combination of based trackers. C1 represents the combination of MEDIANFLOW and MOOSE. C2 represents the combination of MOOSE and KCF. C3 represents the combination of KCF and MIL. C4 represents the combination of MIL, BOOSTING, and CSRT. C5 represents the combination of CSRT, TLD, and MEDIANFLOW. C6 represents the combination of MIL, BOOSTING, CSRT, and TLD. C7 represents the combination of CSRT, MIL, and TLD. 48
5.13 RMSE and Frame Rate of different Proposed Meta-learner. 49
5.14 RMSE Value of proposed Models in the initial and final training process. . . 50

Chapter 1

Introduction

Object tracking is a fundamental aspect of computer vision, enabling the estimation and monitoring of object positions and trajectories in real-time scenarios. Its wide range of applications, including robot vision [26], automated surveillance [24], self-driving [6], defence systems [2], and vision-based control [12], has attracted the attention of researchers from various fields. This process involves identifying and continuously monitoring moving objects in video sequences, sort of like connecting the dots from frame to frame. Accurate object detection is critical because it establishes the groundwork for estimating an object's trajectory over time, which is frequently represented as a series of bounding boxes.

The main tasks of a tracking system are searching and matching. The search process is typically carried out by motion tracking models such as KCF [21], CSRT [32], MIL [13], TLD [52], MedianFlow [14], and MOOSE [15], which capture the object's speed and trajectory and predict future locations based on incoming data. For the matching process, feature extractors and observation models are utilized. Numerous matching techniques have been proposed for this purpose, including template matching, convolutional neural networks (CNNs) [29], LSTM [30], SVM [4], dictionary-based template matching [11], and the mean-shift algorithm [50]. Each of these methods has its advantages and disadvantages, which are discussed in Chapter 2.

Real-time object tracking is a challenging task because of the dynamic nature of the real-time system for example non-linear motion, non-rigid object deformation, blurriness, illumination variation, scale variation, and occlusion. The demand for fast online processing adds another layer of complexity to tracking in dynamic environments. Numerous object tracking algorithms, including BOOSTING [17], KCF [21], CSRT [32], TLD [52], MEDIANLOW [14], MIL [13], MOOSE [15], MDNet [38], DeepSRDCF [48], EBT [53], SRDCF [12], LDP [33], sPST [23], and SC-EBT [43] have been developed for object tracking, but achieving real-time tracking remains a significant challenge. Various traditional trackers, including BOOSTING [17], KCF [21], CSRT [32], TLD [52], MEDIANLOW [14], MIL [13], MOOSE [15], are based on different algorithms, each having their own unique strengths and weaknesses. Some prioritize speed over accuracy, while others are accurate and efficient but

may run more slowly. Considering all these features in mind, Ive developed an innovative real-time visual object-tracking solution that tackles challenges related to scale variation and computational complexity. This research introduces a comprehensive system designed for real-time object tracking that combines traditional tracking methods such as CSRT [32], TLD [52], and MIL [13] with sophisticated meta-learning approaches to intentionally design and improve the precision and efficiency demands of real-time applications. This research emphasizes the significance of using machine learning models as a meta-learner for tracking and as a reliable method for object tracking in dynamic and non-uniform situations. I have tested My proposed technique on the VOT 2015 [27] benchmark dataset and OTB 50 [47] dataset , which outperforms existing deep-learning trackers.

The rest of the thesis is organized as, the previous work in the related domain has been discussed in chapter 2, in chapter 3 and chapter 4 the thesis motivation and proposed method is discussed. Experiment results and conclusion are presented in chapter 5 and chapter 6 respectively.

Chapter 2

Literature Review

Object tracking has multiple roles in multiple types of domains, including security and surveillance systems that use visual data, self-driving cars, traffic management, enhanced medical therapy, customer behavior analysis, interactive game design, and futuristic visual effects. While many algorithms, such as KCF, MIL, CSRT, TLD etc., have been developed for this purpose, but their real-time performance is still frequently lacking. Real-world scenarios present a variety of challenges, including fast motion, nonlinear motion, motion blur, occlusion (partial or full), illumination variation, in-plane rotation, scale variation, and so on, making it difficult to completely depend on a single tracker. Table 2.1 provides an overview of these challenging situations.

Naiyan Wang et al. [28] classified the research conducted for this purpose, which included the cutting-edge visual tracking algorithm, into five distinct components: motion model, feature extractor, observation model, model updater, and postprocessor. Figure 2.1 illustrates a flowchart that represents the basic visual object tracker.

Yilmaz et al. [49] break down object tracking into three categories: point tracking, kernel tracking, and silhouette tracking. The taxonomy of tracking methods is illustrated in Figure 2.2.

2.1 Point Tracking

In computer vision, point tracking involves tracking objects primarily by their point or centroid coordinates while considering their previous states and motion characteristics. This method is categorized into two main approaches: Deterministic and Probabilistic, as proposed by Zhang [51].

The Deterministic approach predicts the object's next position using precise calculations and deterministic models like Kalman filters or template matching. While computationally efficient, deterministic methods may struggle to detect complex motion patterns or occlusions. In contrast, the probabilistic approach incorporates uncertainty into tracking by representing

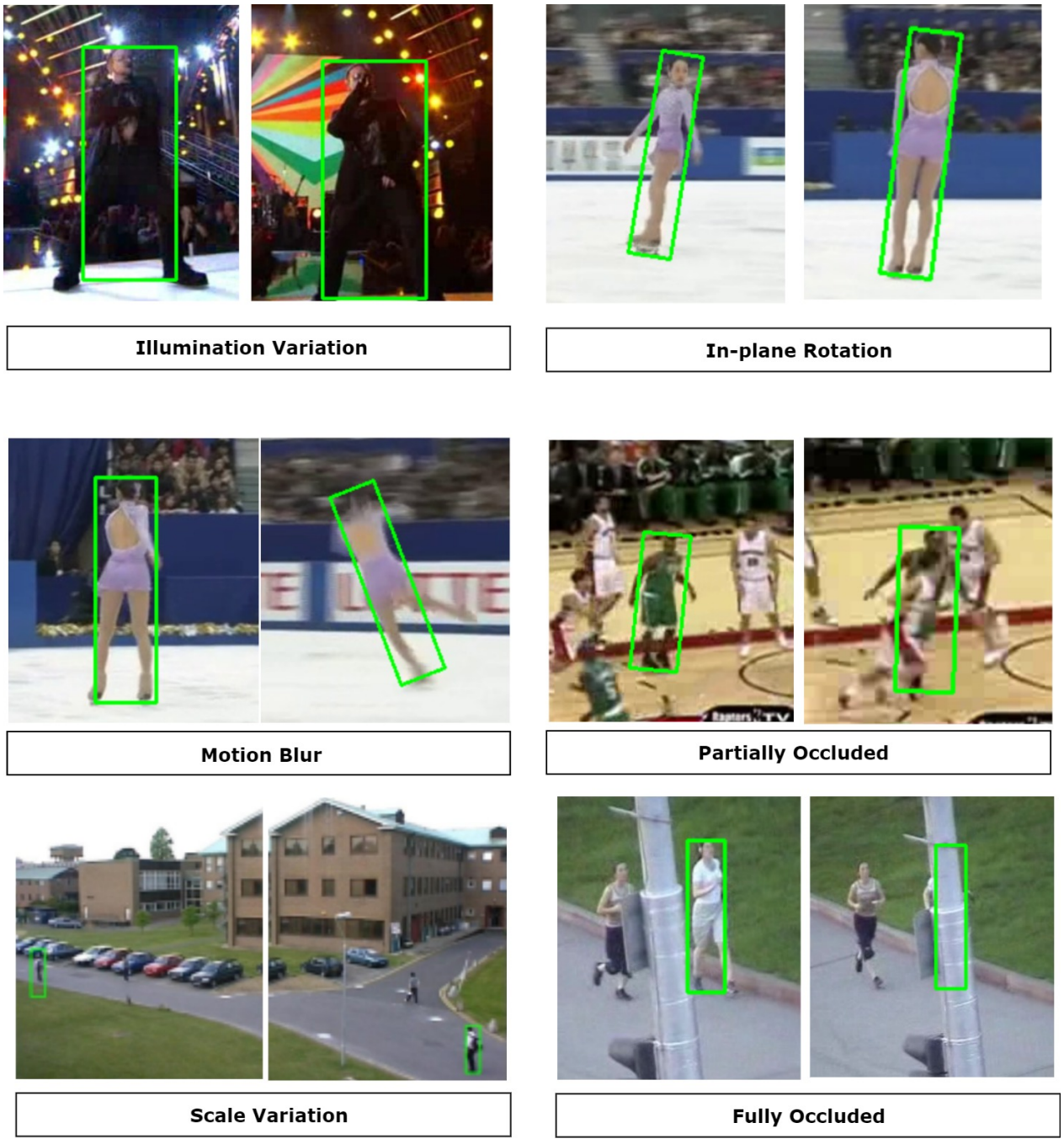


Table 2.1: Different challenging situation

the object's position as a probability distribution. Bayesian filters, for example, deal with uncertainty in measurements and motion dynamics, making them resistant to occlusions and noise. Both have unique capabilities, such as deterministic methods being useful for predicting motion and probabilistic methods excelling at dealing with uncertainty and complex scenarios.

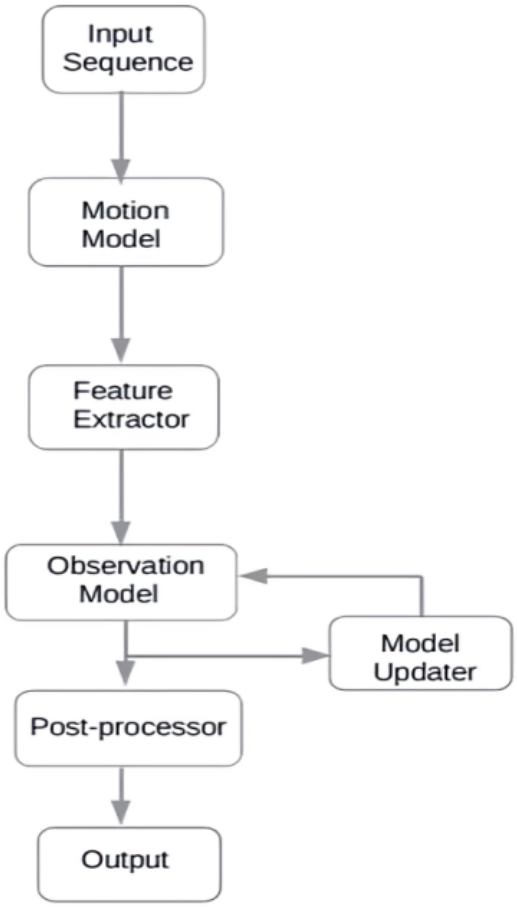


Figure 2.1: Basic flowchart of the visual object tracker

2.2 Kernel Tracking

In the Kernel Tracking method, one or multiple Kernels are defined based on the shape, colour, appearance of the object. The motion of the object is determined by the motion of the kernel which can be rational, translation etc motion. Kernel tracker can be categorized into two types based on the selection of Kernel, Multi-view Based and Template-Based.

Kernels are selected from multiple views or perspectives of the object in Multi-view Based kernel tracking, which allows for robust tracking even when the object's appearance or viewpoint changes significantly. This approach uses information from various perspectives to improve tracking accuracy and robustness. On the other hand, Template-Based kernel tracking is the process of defining kernels based on a single template or representation of the object. While simpler in concept, this approach may fail to track the object when it undergoes significant appearance changes or occlusions. The selecting between Multi-view Based and Template-Based kernel tracking depends on several factors, including the object's appearance variability, tracking scenario complexity, and the computational resources allocated for tracking.

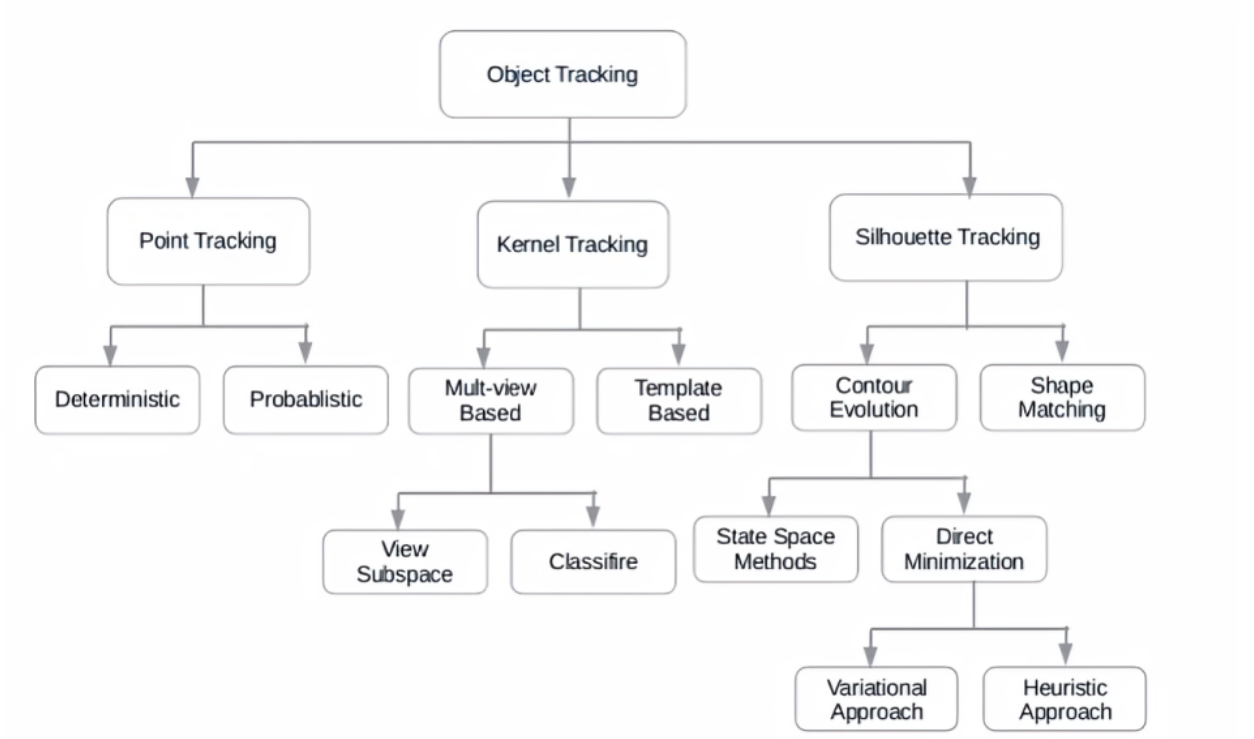


Figure 2.2: Taxonomy of the tracking methods

2.3 Silhouette Tracking

Silhouette Tracking represents objects with greater complexity and accuracy by capturing their precise shapes. This method categorizes into two main types: Contour Evolution and Shape Matching.

Contour Evolution in Silhouette Tracking refers to the continuous refinement of an object's boundary contour over multiple frames. Techniques such as active contours and level set methods are used to allow the contour to evolve and align with the object silhouette. This allows for accurate tracking even in challenging scenarios involving deformations or occlusions. Shape Matching, on the other hand, is concerned with matching the extracted silhouette of an object to predefined shape templates or models. The method determines the best match between the object's silhouette and the template by comparing features such as curvature, angles, and moments, allowing for robust tracking in different conditions. The decision between Contour Evolution and Shape Matching in Silhouette Tracking is determined by factors such as the object's deformation characteristics, the presence of occlusions, and the computational complexity acceptable for the tracking application. Each approach has distinct advantages in capturing and tracking objects with complex shapes and dynamics.

Examples of all types of trackers and their examples are shown in Table 2.2.

Category	Subcategory	Example(s)
Point Tracking	Deterministic approach	Mean Shift Algorithm, MGE tracker
	Probabilistic approach	Kalman Filter, Particle Filter, Unscented Kalman Filter
Kernel Tracking	Multi-view Based	SVM
	Template Base	Mean Shift Algorithm, KLT, Template Matching
Silhouette Tracking	Contour Evolution	State space models, Heuristic methods, Contour tracking
	Shape Matching	Hough transform, Histogram

Table 2.2: Different Categories of Tracking and Their Examples

Feature selection is the most important and difficult task in the object tracking process. Different features mainly represent the object to the tracker model. Therefore a feature extractor is one of the main components of an object tracker. With respect to feature extractor trackers can be categorized into two types, the Generative method and the Discriminative method. Different features, their descriptor and related works are listed in Table 2.3

2.4 Generative method

The Generative model is based on the probability density of an object's appearance in the next frame, which is calculated using extracted features. Thus, feature extraction plays a vital role in this tracking method. Illustrated in Figure 2.3, the framework of the Generative model elucidates its operation. Common generative models include density estimation [19], Gaussian mixed models [34], dictionary learning [16] [44], hidden Markov models [39], and regression networks [1]. These models use a variety of methods to estimate the probability distribution of an object's appearance, resulting in accurate tracking. Generative models, which rely on extracted features, provide a versatile approach to object tracking, with applications ranging from surveillance to medical imaging.

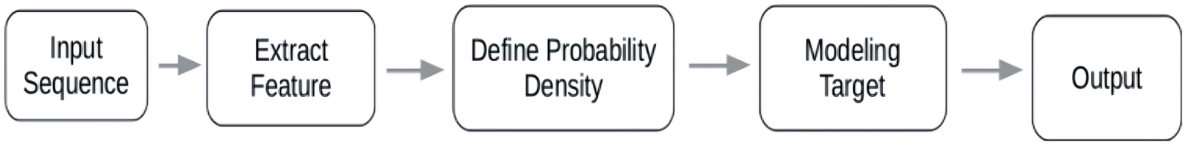


Figure 2.3: Flow chart of Generative models

2.5 Discriminative method

The Discriminative model operates independently of data generation methods, focusing solely on determining the error between object and background. It is primarily involved in binary classification tasks, distinguishing between objects and backgrounds. At each step, the model learns to distinguish between the two. This tracking method involves comparing the object's similarity between consecutive frames and detecting it in the current frame, also known as pseudo-tracking or Tracking-by-Detection. Figure 2.4 illustrates the architecture of discriminative models. Multiple instance learning (MIL) [13], structured output SVM [4], boosting

techniques [18], ensemble classifiers [5] [45] [3] [37], and P-N learning [25], TLD [52] are some common examples. These models use a variety of strategies to achieve accurate object detection and tracking, which proves useful in tasks such as surveillance and image analysis.

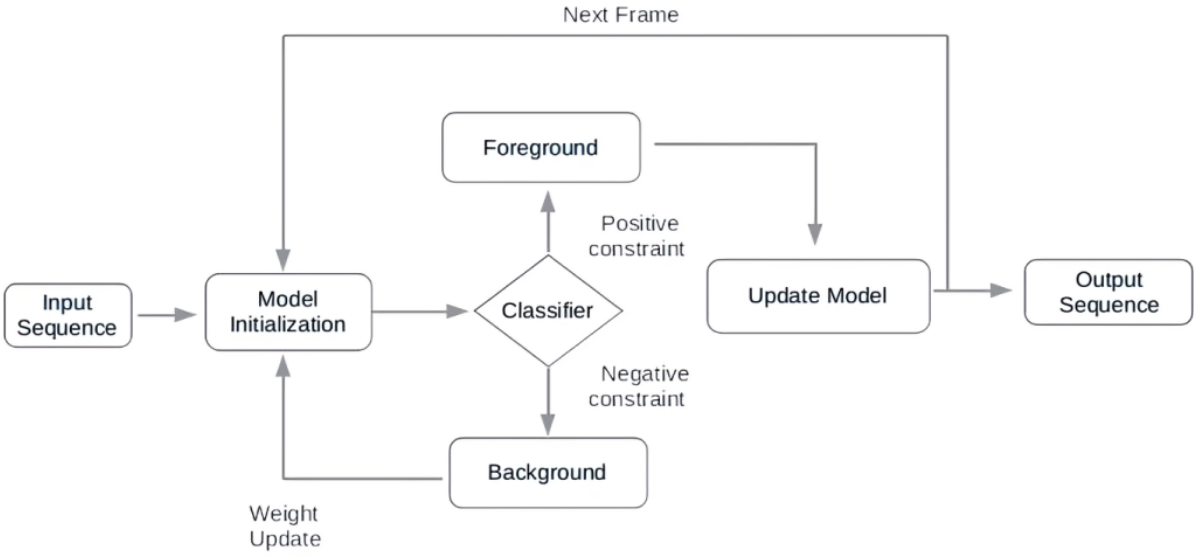


Figure 2.4: Flow chart of Discriminative models

These days the two most important Discriminative models are the Correlation model and the Deep learning model.

2.5.1 Correlation model

Correlation refers to the convolution of two signals in the field of signal processing, aimed to find out whether the correlation exists or not.

In Computer Vision,this technique is to find out the exact location where the correlation exists. If an image is defined by f and the template is denoted by h then the correlation is given by

$$(f \otimes h)(t) = \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau \tag{2.1}$$

$\int$ use for continuous signals, however, as the image pixels are discrete. Equation (2) is use for correlation calculation, which is

$$(f \otimes h)(n) = \sum_{N=-\infty}^{+\infty} f(N)h(N+n) \tag{2.2}$$

One of the major problems in the correlation filter is time complexity, which is $O(n^2)$. To overcome this problem, correlations are calculated in the frequency domain by simply transferring Fast Fourier Transform (FFT) or Discrete Fourier Transform (DFT). After finding

Feature	Feature Descriptor	Advantage	Disadvantage	Example
Color	CN, CBP, CC	Best for robustness to photometric changes	Cannot handle color change	DAT [35], ASMS [41],
Haar-like	Haar-like, Hairlike extra	Very efficient and computation time does not depend on the size of the object.	Complex	MIL [13], BOOSTING [17]
Histogram Gradient Feature	HOG, SHFT, ORB	Immune to geometric and optical deflection	-	CSR-DCF [31], KCF [21]
Texture Feature	Gabor wave, LBP, WLD	Invariance in grayscale and rotation, accuracy is good	Expensive computation	TLD [52]
Grayscale	IH, HOI, DF	Easy to implement, Need low computation power	Sensitive to lightning changes.	Frag [42]
Deep Feature	Conv. feat	Best feature extractor for state-of-the-art model	Expensive computation, Some time need GPU	SiameseFC [7], CFNet [40]
Motion	Optical Flow	Good for occlusion handle	Expensive computation	MedianFlow [14]
Multiple Feature	-	Improve robustness and accuracy	Expensive computation	HOG LBP [46], TOFF [36]
Spatiotemporal information	Super-voxel, Super-pixel	Pose change and partiallyoccluded object	Cannot track fully occluded object	-
Shape	-	Low computational time	-	-
Depth	3D	Immune to drift and occlusion	-	[9]
Contour	Edge	Less sensitive to illumination changes	-	-

Table 2.3: Different features, Features Descriptor and their Advantage, Disadvantage and Example

the correlation, all the images are transferred back to the time domain by applying Inverse Fast Fourier Transform (IFFT) or Inverse Discrete Fourier Transform (IDFT). This makes the computation more effective with time complexity $O(n \log n)$.

The general framework for correlation model visual object tracking methods, as summarized by Chen et al. [10], is shown in Fig 2.5. Examples of the most common correlation models include:

- Minimum Output sum of Squared Error (MOSSE) [15]
- Correlation Filter-based Trackers (CFT) [20]
- Average of Synthetic Exact Filters (ASEF) [8]
- Kernel Correlation Filter (KCF) [21]
- Tracking-Learning-Detection (TLD) [52] [P-N learning [25]]
- Collaborative Correlation Tracker (CCT)

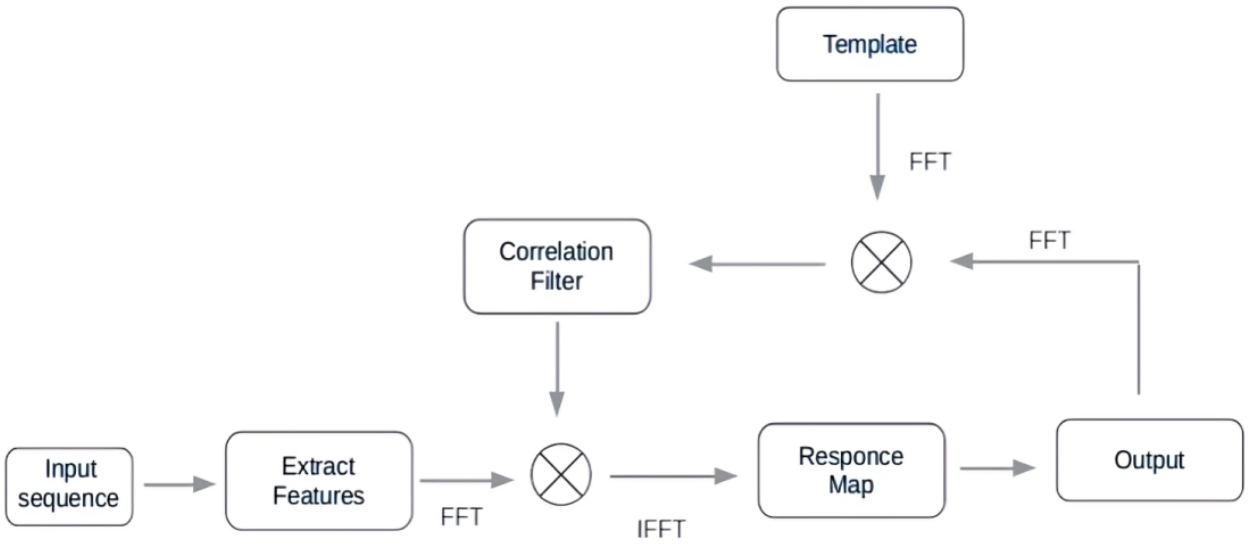


Figure 2.5: A general framework for correlation model

2.5.2 Deep learning model

In the field of computer vision, Deep Learning (DL) based object tracking has become increasingly popular in recent years. However, the implementation of deep learning models requires significant computational power typically found in GPUs, resulting in limited portability and challenges in achieving real-time performance. Nevertheless, these trackers demonstrate superior accuracy compared to other tracking methods due to their strong decision-making capabilities. Deep learning-based trackers are categorized into three main types: CNN-based, RNN-based, and Discriminative deep learning tracker, as illustrated in figure 2.6.

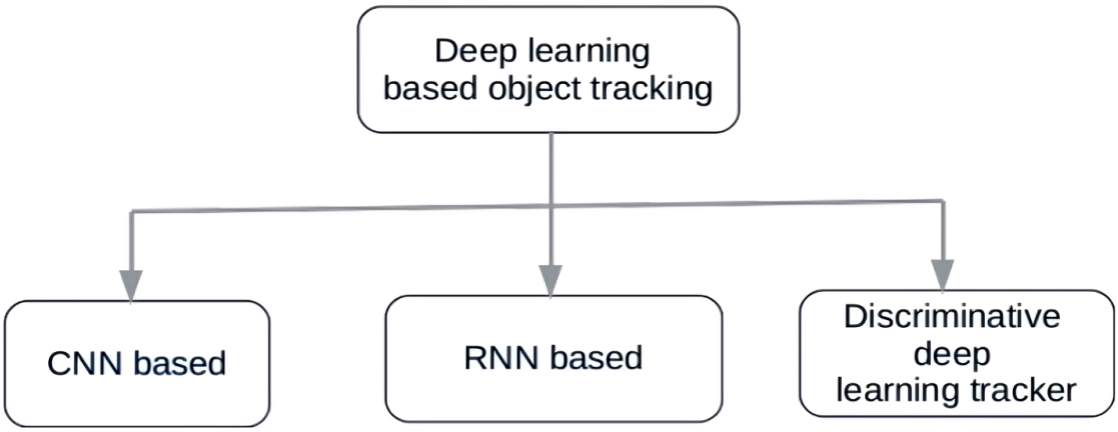


Figure 2.6: Classification Of Deep learning-based Object tracking

CNN based

CNN (Convolutional Neural Network) is a powerful tool in deep learning, particularly for image classification. Its strength comes from its ability to automatically extract meaningful features from images. CNN-based algorithms, on the other hand, require a significant amount of computational resources because CNN layers act as both feature extractors and observation models. This high computational requirement reduces their efficiency, especially on hardware like ARM processors commonly found in mobile devices.

CNN-SVM [22] is one of the first deep learning projects proposed by the Korean POSTECH team, combining CNN with Support Vector Machines (SVM) for tracking purposes. While CNN-based tracking is more accurate than other methods, it has limited hardware efficiency and frequently relies on servers or desktop computers due to its computational complexity.

To tackle this challenge, researchers have proposed hybrid architectures that combine the CNN architecture with Correlation Filters (CF) to enhance tracking speed. For instance, Bertinetto et al.'s CFNet [40] blends CNN and CF to expedite tracking, albeit with a slight compromise on accuracy. Remarkably, advancements like DCFNet 2.0 have achieved impressive tracking speeds of 100 frames per second (FPS) on Graphics Processing Units (GPUs), albeit often sacrificing accuracy. The Siamese architecture has become a popular choice for deep learning-based trackers. Siamese architecture involves convolution of template and input frames using the same network. The final layer is connected to a correlation filter, which determines the object's position and correlation. Figure 2.7 represents the Siamese architecture-based deep learning tracking framework.

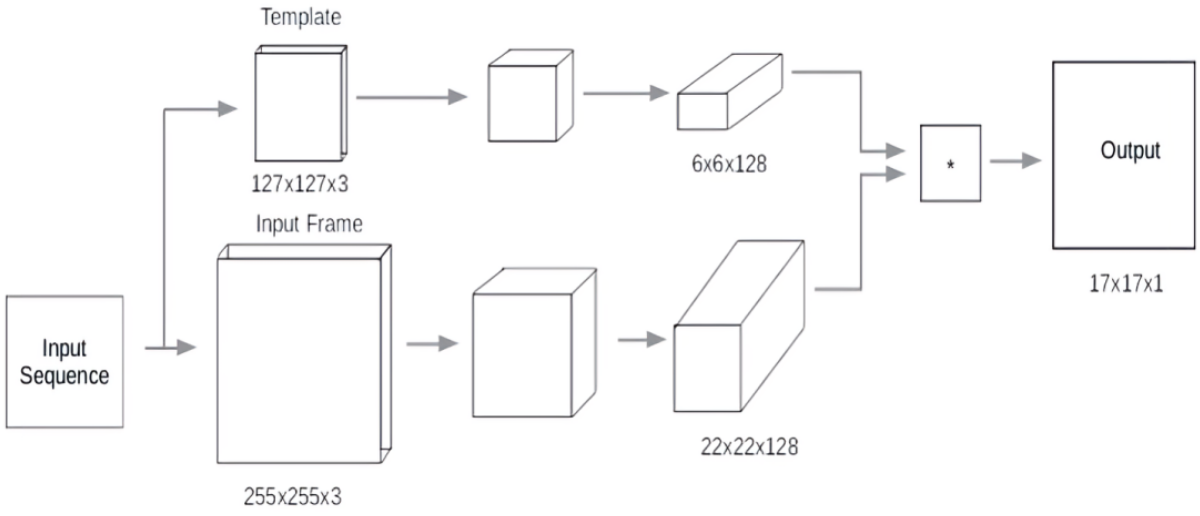


Figure 2.7: Siamese architecture based Deep learning model

RNN based

Recurrent Neural Networks (RNNs), also known as Long Short-Term Memory (LSTM), have the unique ability to retain information from previous data, whereas CNN-based models rely solely on the current input frame. The ability to include both current and previous input frames improves the efficiency of tracking algorithms. For example, Gan et al. proposed a novel visual tracking approach based on RNN, whereas Krishnan et al. created an efficient tracker by combining RNN with the Kalman Filter, a method for estimating the state of a dynamic system. Held et al. presented an RNN-based tracker known for its high frame rate and ability to handle occlusions, which occur when objects are temporarily obscured from view. Furthermore, Haarnoja et al. used the Kalman Filter in the backpropagation process of RNN tracking models, which helped to improve performance. Overall, RNN-based trackers take advantage of their ability to retain historical data, allowing them to effectively track objects across frames and handle complex tracking scenarios.

Discriminative deep learning tracker

Despite the success of CNN and RNN-based trackers in achieving high tracking accuracy, researchers are exploring alternative approaches, such as Discriminative Deep Learning Tracker (DLT) techniques. DLT tracks objects by directly cropping images or frames, which simplifies the tracking process. Wang et al. presented an efficient and robust DLT method, demonstrating its usefulness in object tracking tasks. Building on this, Zhuang et al. proposed a discriminative classifier with a shallow and deep architecture-based tracker, which is more efficient than traditional DLT approaches.

Zhuang et al.'s tracker combines the advantages of shallow and deep architectures, using discriminative classifiers to improve tracking performance. This hybrid approach simplifies the tracking process while maintaining high accuracy. By incorporating both shallow and deep architecture components, the tracker improves efficiency, making it a promising solution for real-world tracking applications.

Overall, Discriminative Deep Learning Tracker methods offer an alternative framework in object tracking, that emphasise simplicity and efficiency. Wang et al. and Zhuang et al.'s contributions demonstrate the potential of such methods for enhancing existing CNN and RNN-based trackers, providing a range of options for addressing tracking challenges with varying computational resources and requirements.

2.6 Different Motion Models in Object Tracking

The tracker motion model or dynamic model can determine how an object moves. Which can be categorized into two types, linear motion model and nonlinear motion model.

2.6.1 Linear motion model

In a tracking system, the velocity of an object can be estimated using regression techniques, enabling real-time motion analysis based on previous object movements. Andriyenko et al. demonstrated a tracker capable of associating velocity with each object detection, facilitating accurate velocity estimation during tracking tasks. Additionally, Collins introduced a linear motion-based tracker in which objects are tracked with a constant velocity, providing a simplified yet effective method for motion analysis.

2.6.2 Nonlinear motion model

In real-time scenarios, a simple linear motion model may not be sufficient to accurately predict an object's exact position. That's where nonlinear motion models come in, providing more accurate predictions. Researchers such as Dicle et al. and Yang et al. have proposed robust multi-object trackers based on these nonlinear models. On the other hand, Yoon et al. proposed using a linear motion model based on Relative Motion Network (RMN) camera motion for robust tracking.

2.7 Different selection based trackers

Representative selection based trackers are categorized into four types, **Dictionary-based trackers, Ensemble-based trackers, Segmentation-based trackers and Circulant trackers**.

2.7.1 Dictionary-based trackers

Dictionary-based trackers utilise a dictionary that contained target templates of objects. These trackers use this dictionary to predict the location of objects in new frames. Xing et al. proposed combining long-term, mid-term, and short-term dictionaries to improve tracking accuracy and robustness. Similarly, Wang et al. developed a robust tracking system based on dictionaries. They proposed a modification to improve its performance further.

2.7.2 Ensemble-based trackers

Ensemble techniques are powerful methods that combine the predictions of multiple weak classifiers to make strong classifications of objects. Cao and Xue introduced an adaptive random forest approach, which maintains a collection of decision trees for target objects. This method uses these trees to ensemble the objects and make predictions. Similarly, Bai et al. suggested combining weak classifiers trained on 8x8 patches within the object's bounding box. They used weight vectors sampled from a Dirichlet distribution, which are updated over time to improve accuracy. Wang et al. proposed a novel approach that combines ensembles of different trackers using a conditional particle filter method.

2.7.3 Segmentation-based trackers

Segmentation involves dividing images into similar regions, employing various algorithms such as Mean Shift Clustering, Graph-Cuts, Normalized Cuts, and Active Contours. Yilmaz et al. discuss different segmentation techniques utilized in tracking systems. Duffner et al. introduced PixelTrack, which combines a co-trained probabilistic segmentation model with a pixel-based Hough model to address object deformation. Segmentation methods can be broadly categorized into Motion Segmentation and Foreground Segmentation.

Motion Segmentation

Motion Segmentation is commonly applied in surveillance tracking. In tracking sequences, the majority of pixels remain static while target objects exhibit dynamic behavior. This tracking model is advantageous due to its effectiveness and simplicity in approximating moving objects.

Foreground Segmentation

Foreground segmentation classifies pixels in an image into two categories: foreground and background. In this tracker model, static objects like buildings and trees are classified as Background, whereas objects that change over time are classified as Foreground. This consideration is taken into account during the tracker's development.

Circulant trackers

Circulant trackers leverage the circulant structure of adjacent subwindows of templates to create an exceptionally fast-tracking system. Danelljan et al. introduced innovative attributes to enhance performance on challenging sequences of frames. Henriques et al. developed a state-of-the-art KCF model utilizing Gaussian kernel and histogram-of-oriented-gradients (HOG) features, achieving a remarkable speed of 172 frames per second (FPS). Yao et al. presented a part-based tracker employing online learning with structured SVM. Shen et al. proposed a versatile boosting framework for integrating weak structured online learning trackers.

2.8 Offline and Online Object Tracking

Based on the way of trajectory generation, object trackers can be classified in two types, Offline object tracking and Online object tracking.

2.8.1 Offline object trackin

In this type of tracker, the model is provided with the entire video sequence. Offline object trackers build a graph that represents the tracking problem. Various methods can be employed to solve this graph, such as K-shortest Path, Sub-Graph Decomposition based on Multi-Cut, Min-Cost Network Flow, and Multi-Clique.

2.8.2 Online object tracking

In offline object tracking models, all frames are provided to the model. Trackers predict the next location of the object by computing the joint probability of the features and object location. This probability distribution can be either Gaussian or non-Gaussian in nature. Zhou et al. utilized a Gaussian mixture probability hypothesis density filter to estimate the target object's location. The Kalman filter-based tracker is among the most widely used Gaussian trackers. Particle filter trackers are the most popular non-Gaussian trackers. Various matching costs, such as those based on the Markov decision process, random forests, and energy minimization, are employed for object tracking.

According to time accession tracker can be categorized into two types, Real-Time tracker and Non-Real-Time tracker. Examples of some real-time and non-real-time object trackers are shown in Table 2.4.

Table 2.4: Example of real-time and non-real-time tracking and their implementation method

Tracker	**Method**	**Real-Time**
KCF	Correlation Filter	Y
OAB	Integrated learning	Y
CSRT	Correlation Filter	Y
MIL	Online learning	Y
MOSSE	Correlation Filter	Y
TLD	PN-Learning	Y
UKF8	Probabilistic approach	Y
Mean Shift	Kernel Tracking	Y
Particle Filter	Probabilistic approach	Y
SANet	Deep learning	N
TCNN	Deep learning	N
PTAV	Deep learning	Y
SiamRPN	Deep learning	Y

Chapter 3

THESIS MOTIVATION AND CONTRIBUTION

Various traditional object trackers, such as BOOSTING, KCF [21], CSRT [32], TLD [52], MEDIANLOW [14], MIL [13], and MOOSE [15], rely on diverse algorithms, each presenting distinct strengths and weaknesses. Some prioritize speed over accuracy, while others aim for a balance between accuracy and efficiency, albeit at the expense of speed. In light of these considerations, we have developed an innovative real-time visual object-tracking solution to address challenges related to scale variation and computational complexity.

This research presents a comprehensive system designed for real-time object tracking, which integrates traditional tracking methods like CSRT, TLD, and MIL with advanced meta-learning approaches. My objective is to intentionally improve precision and efficiency to effectively meet the demands of real-time applications. Central to my research is the recognition of the significance of employing machine learning models as meta-learners for tracking. These models offer a reliable method for object tracking in dynamic and non-uniform scenarios.

My proposed technique has been rigorously tested on OTB 50 and VOT 2015 benchmark dataset. The results demonstrate that it outperforms existing deep-learning trackers. My approach enhances performance by integrating the strengths of traditional tracking algorithms with meta-learning strategies, particularly in scenarios involving scale variation and computational complexity. This study advances the field of real-time object tracking by providing a robust solution capable of meeting the changing demands of various applications in computer vision and beyond.

Chapter 4

Proposed Methodology

In this research work, we introduce a meta-learning strategy for enhancing object tracking performance, which integrates established trackers such as BOOSTING [17], KCF[21], CSRT [32], TLD [52], MEDIANLOW [14], MIL [13] and MOOSE [15] with a Regression-based Machine Learning Model. The proposed algorithm is described in Algorithm 1. The initial phase involves the independent operation of traditional trackers to predict the target object's location, offering diversity and adaptability across various tracking conditions. The input frame and the target object are given to the base tracker independently, which makes the algorithm more robust in different challenging situations. After that, we employ a meta-learner to refine the predicted coordinates by integrating the strengths of individual trackers. The meta-learner takes the concatenated output of the base tracker, which is $[\tilde{P}^1_{tn}, \tilde{P}^2_{tn}, ..., \tilde{P}^N_{tn}]$. In this research work, we use a machine learning-based regression model as the meta-learner, as we have to find the coordinates of the target.

Adding the regression model improves overall tracking accuracy, especially in tough situations like when objects are blocked, change in size, or move suddenly. This proposed methodology employs the meta-learning principle, where the model learns from the diverse outputs of traditional trackers and adapts its predictions accordingly. This adaptive nature allows the system to perform effectively and robustly under various tracking conditions. To track the target more accurately, we also used the previous frame output as the input for the meta-learner, so that it has some idea about the previous output, which leads to better accuracy of the whole tracking system.

We select the base tracker according to the RMSE between the predicted value and the ground truth and processing time, as shown in Table 4.1. We select the base tracker in such a way that it can well balance both accuracy and processing time. The schematic diagram of the proposed model is shown in Figure 4.1.

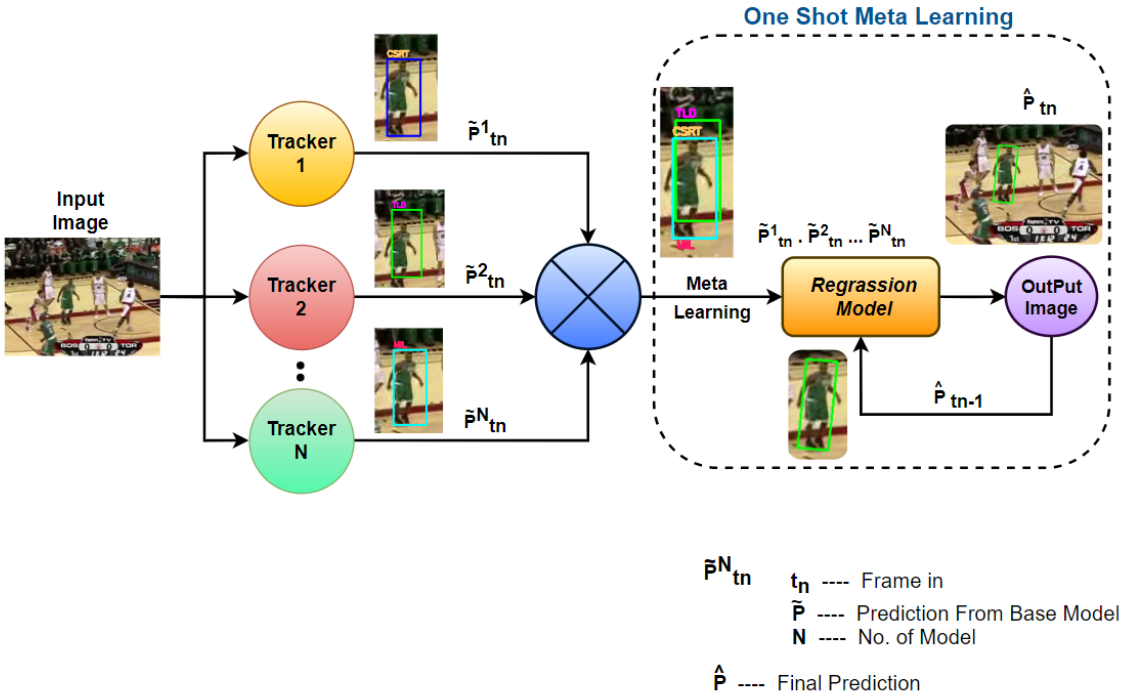


Figure 4.1: Flowchart of the one-shot meta-learning system: The tracker stores previous tasks, guides meta-learning for regression model updates, and enables learning from the predicted output.

Algorithm 1

Require: Object template, Input Frame
Ensure: Predicted center location of the target (x, y), Estimated object size (h, w)

Initialization:
Initialized base trackers
Initialized meta-learner
Initialized Template of the target
Position Estimation:
Calculate the target position using base trackers $P_{\sim N_{t_n}}$
Concatenate the predicted position $[P_{\sim 1_{t_n}}, P_{\sim 2_{t_n}}, ..., P_{\sim N_{t_n}}]$
if $FrameNumber \neq 1$ **then**
    Concatenate predicted position given to the meta-learner
else
    Previous frame object position and Concatenate predicted position given to the meta-learner
end if

4.1 Experimental Setup:

The model has been trained and tested on a Google Collab virtual machine equipped with Python 3.8.1 and Python OpenCV 4.8.0.76 library and a system configuration of Intel Core i5, 1.60 GHz CPU with 8GB of RAM. Multiple video sequences with different challenging conditions are tested with the proposed tracker. On the Visual Tracker Benchmark, the proposed method is compared to fourteen existing tracking models, including BOOSTING [17], KCF[21], CSRT [32], TLD [52], MEDIANLOW [14], MIL [13], MOOSE [15], MDNet [38], DeepSRDCF [48], EBT [53], SRDCF [12], LDP [33], sPST[23] and SC-EBT[43], with different properties such as Accuracy, Robustness, Speed, Adaptability, Handling of Occlusions, and Real-time Performance. The tracking outcomes are evaluated using standard qualitative and quantitative approaches, such as precision metrics, object tracking error, average expected overlap (EAO), and success rate. These evaluations provide an in-depth analysis of tracking performance, combining both quantitative and qualitative methodologies.

4.2 Data-set Description:

To maintain high academic standards, we need to use benchmark datasets to evaluate the proposed object tracking algorithm. In this thesis, we have diligently chosen two well-known and widely used benchmark datasets in the field of object tracking: the Visual Object Tracking (VOT) Challenges dataset and the OTB-50 (Object Tracking Benchmark) dataset.

The proposed model's performance is evaluated using video sequences from the OTB50 and VOT 2015 Challenges datasets. The OTB50 dataset is composed of 50 short videos, each showcasing a variety of challenging tracking scenarios. In contrast, the VOT 2015 dataset contains 60 videos that have been carefully curated by combining information from various sources, including the ALOV dataset, the OTB100 dataset, and other datasets that focus on specific tracking challenges.

These datasets offer meticulously annotated video sequences, accompanied by ground truth information regarding the precise object locations. The primary differentiation between the two datasets lies in the representation of the ground truth values. In the VOT 2015 dataset, the ground truth values are provided in the form of poly-lines, while in the OTB dataset, they are represented as rectangular shapes.

These datasets contain videos with various challenges, such as fast movement, obstructions, shape changes, rotations, and complex backgrounds. This diversity allows for a detailed and complete assessment of object tracking algorithms, helping researchers determine their performance and versatility in different difficult tracking situations.

Table 4.1: RMSE Value and Processing Time of different individual Based Tracker

Based Model	RMSE	Frame Rate (in second)
MIL [13]	132	21.18
BOOSTING [17]	150	29.15
CSRT[32]	171	20.10
TLD[52]	179	15.30
MEDIANFLOW[14]	222	513.82
MOOSE [15]	272	2106.82
KCF[21]	276	204.74

4.3 Training Process :

The root mean squared error (RMSE) is calculated as the square root of the mean of the square of all errors. RMSE is widely used and considered as an excellent general-purpose error metric for numerical predictions.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} S_i^2 - O_i^2} \tag{4.1}$$

The first column of the Table 4.1 shows that MIL, BOOSTING, CSRT, and TLD trackers perform better results than MEDIANFLOW, MOOSE, and KCF in terms of RMSE value. And the second column represents the Frame Rate of every based model, which is basically number of frames process per unit of time. Based on that result, we select the base model that initially predicted the bounding box in the first training. As shown in this Table 4.1, it basically indicate that, MOOSE, KCF and MEDIANFLOW gives low accuracy but fast processing speed. On the other hand, MIL, BOOSTING, CSRT and TLD give high accuracy but low processing speed.

The experimental results in Figure 4.1 show how My meta-learning technique improves object tracking in computer vision applications. By utilizing meta-learning, My model not only improves tracking accuracy but also provides a dependable and adaptable solution to real-world tracking challenges. This research opens up new door for the development of more sophisticated and resilient object-tracking systems in diverse environments at very minimum processing time.

Figure 4.2 illustrates My training procedure. To ensure diversity, we first collect predicted coordinates from based trackers. Then we apply these coordinates as an input of a regression model to improves accuracy constantly in challenging tracking scenarios.

We employ traditional trackers such as BOOSTING, KCF, CSRT, MIL, TLD, MedianFlow, and Moose in My methodology. The specific combination of these trackers depends on the desired balance between accuracy and processing speed. For instance, if prioritizing higher accuracy, we opt for MIL, CSRT, BOOSTING, and TLD trackers as the base trackers. Conversely, if faster processing speed is the priority, we select KCF, MedianFlow, and Moose trackers.

In this thesis, we analyze seven distinct combinations of base trackers. Among these, the combination of CSRT, MIL, and TLD trackers is used to prioritize accuracy over processing speed.

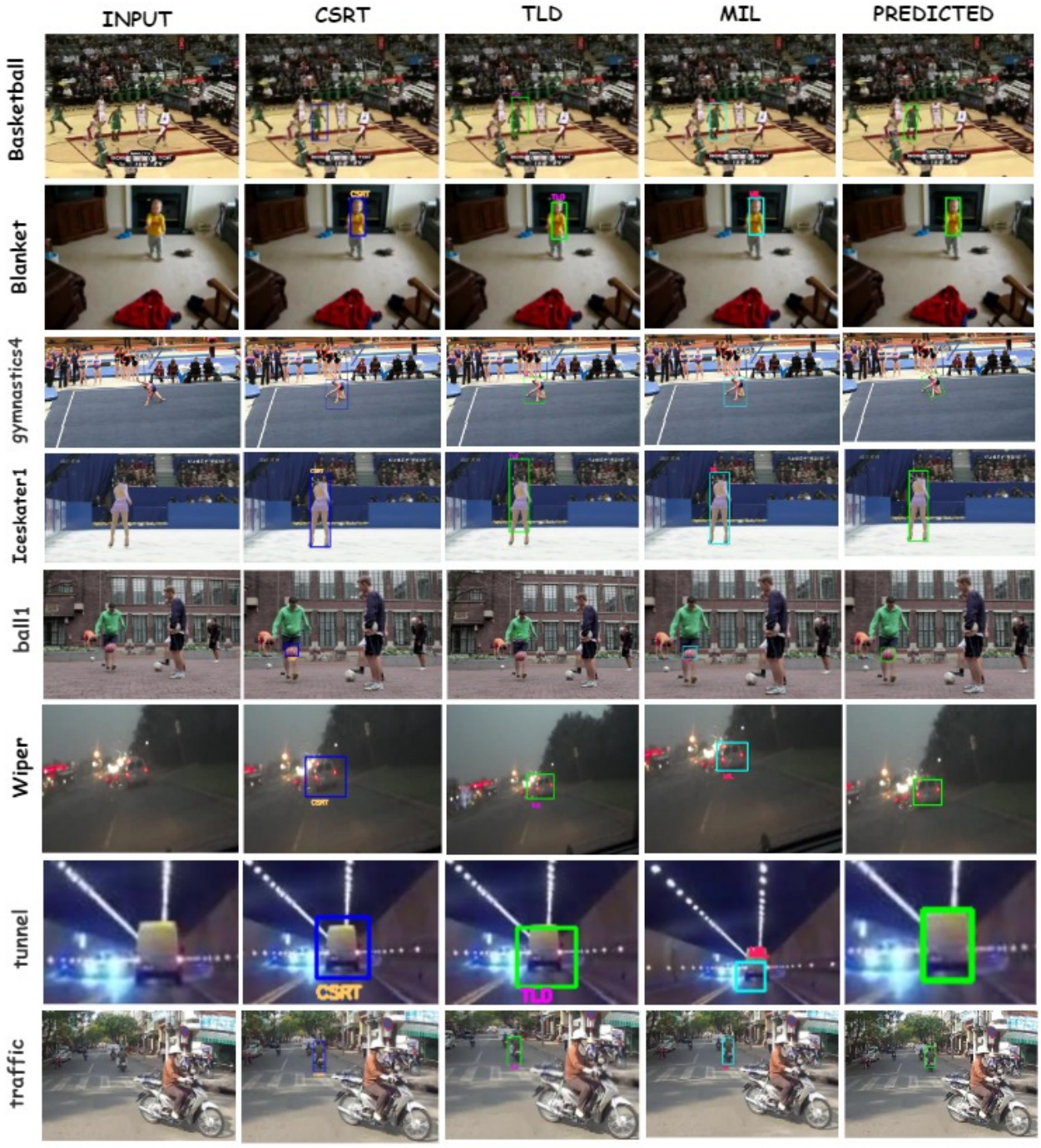


Figure 4.2: Illustrates the process of proposed tracking on some videos taken from Visual Object tracking benchmark dataset, VOT2015 [27].

Chapter 5

Result Analysis

5.1 Qualitative Comparison :

A thorough qualitative assessment of the proposed model is carried out by analyzing the Precision Plot, Center Location Error, Success Plot, and tracking results using bounded boxes. This evaluation focuses on nine challenging video properties: Illumination Variation (IV), Non-rigid Object Deformation (DEF), Partial or Full Occlusion (OCC), Low Resolution (LR), Fast Motion (FM), Motion Blur (MB), Scale Variation (SV), Background Clutter (BC), and In-Plane Rotation (IPR). The aim of this analysis is to provide an understanding of the model's performance under diverse conditions and to evaluate its robustness and effectiveness across various tracking challenges shown in table 5.1.

Attribute	**Description**
IV	Illumination in the target region is changed.
DEF	Non-rigid Object Deformation.
OCC	Target is partially or fully occluded.
LR	The number of pixels inside the ground-truth bounding box is less than tr (tr=400). tr = threshold of number of pixels.
FM	The motion of the object is larger than tm pixels (tm=20). tm = motion threshold.
MB	The target region is blurred due to the motion of the camera
SV	The average ratio of the bounding box size in the first frame to the bounding box size in the current frame falls outside the range of [1/ts, ts], where ts is the scale threshold, and ts is set to 2.
BC	The background near the target has a similar color or texture as the target.
IPR	Target rotates in the image plane.

Table 5.1: Different attribute description

The characteristics of the videos utilized in this thesis are presented in Table 5.2. For the purpose of qualitative comparison, the Precision Plot, Center Location Error, Success Plot, and Tracking Results by Bounding Boxes methods are employed.

Video sequences name	**Number of frames**	**Object Size**	**Challenging conditions**
Basketball	724	34 x 81	IV, OCC, DEF, OPR, BC
Biker	142	16 x 26	SV, OCC, MB, FM, OPR, OV, LR
Bird2	98	69 x 73	OCC, DEF, FM, IPR, OPR
BlurBody	334	87 x 319	SV, DEF, MB, FM, IPR
BlurCar2	583	122 x 99	SV, MB, FM
Boy	602	35 x 42	SV, MB, FM, IPR, OPR
Bolt	348	26 x 61	OCC, DEF, IPR, OPR
Bolt2	291	34 x 64	DEF, BC
Car1	1018	66 x 55	IV, SV, MB, FM, BC, LR
Car4	659	107 x 87	IV, SV
Car24	3057	27 x 24	IV, SV, BC
CarDark	391	29 x 23	IV, BC
CarScale	250	42 x 26	SV, OCC, FM, IPR, OPR
Crowds	347	22 x 51	IV, DEF, BC
ClifBar	470	30 x 54	SV, OCC, MB, FM, IPR, OV, BC
David	469	64 x 78	IV, SV, OCC, DEF, MB, IPR, OPR
Dancer	225	47 x 102	SV, DEF, IPR, OPR
Deer	71	95 x 65	MB, FM, IPR, BC, LR
DragonBaby	113	56 x 65	SV, OCC, MB, FM, IPR, OV
Freeman4	281	15 x 16	SV, OCC, IPR, OPR
FaceOcc1	892	114 x 162	OCC
Girl	498	31 x 45	SV, OCC, IPR, OPR
Girl2	1498	44 x 171	SV, OCC, DEF, MB, OPR
Human2	1126	95 x 325	IV, SV, MB, OPR
Human4	665	27 x 82	IV, SV, OCC, DEF
Human6	790	18 x 55	SV, OCC, DEF, FM, OPR, OV
Human8	126	30 x 91	IV, SV, DEF
Ironman	166	49 x 57	IV, SV, OCC, MB, FM, IPR, OPR, BC, LR
Jump	120	52 x 182	SV, OCC, DEF, MB, FM, IPR, OPR
Jogging	307	25 x 101	OCC, DEF, OPR
Jogging	307	37 x 114	OCC, DEF, OPR
Jumping	313	34 x 33	MB, FM
Liquor	1739	256 x 152	IV, SV, OCC, MB, FM, OPR, OV, BC
Lemming	1334	61 x 103	IV, SV, OCC, FM, OPR, OV
Man	134	26 x 39	IV
Matrix	98	38 x 42	IV, SV, OCC, FM, IPR, OPR, BC
MotorRolling	162	122 x 125	IV, SV, MB, FM, IPR, BC, LR
Rubik	1997	73 x 74	SV, OCC, IPR, OPR
Skating2	471	64 x 236	SV, OCC, DEF, FM, OPR
Skater	158	39 x 137	SV, DEF, IPR, OPR
Tiger2	352	76 x 84	IV, OCC, DEF, MB, FM, IPR, OPR, OV
Twinings	472	74 x 55	SV, OPR
Woman	597	21 x 95	IV, SV, OCC, DEF, MB, FM, OPR
Walking	412	24 x 79	SV, OCC, DEF
Walking2	500	31 x 115	SV, OCC, LR

Table 5.2: Object properties used in this thesis

5.1.1 Success Plot :

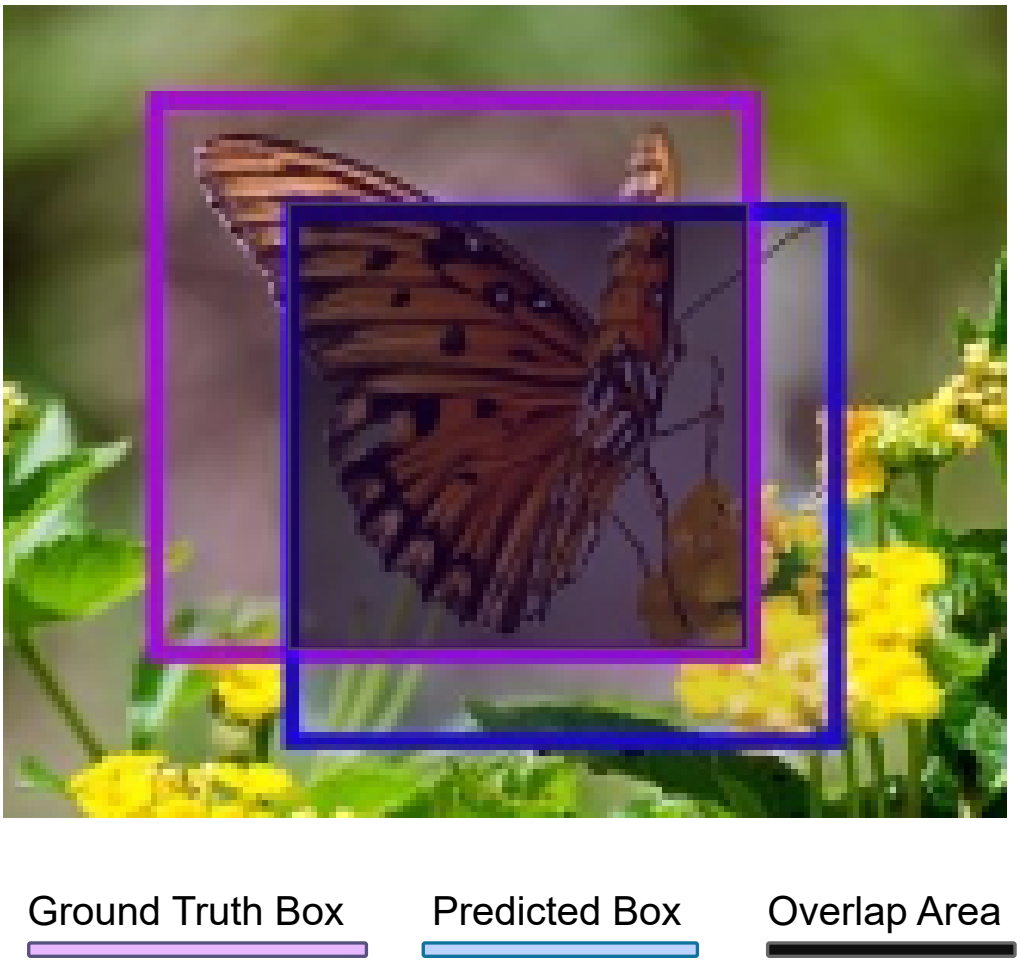


Figure 5.1: An illustration of Success Plot

Intersection Over Union (IoU) serves as a crucial metric for assessing the accuracy of object tracking models. The IoU is computed by determining the overlap percentage between the actual object's location and the prediction made by the tracking model. This computation is performed for every frame concerning the ground truth and the predicted bounding box. IoU is given by the equation

$$IoU = \frac{R_t \cup R_g}{R_t \cap R_g} \tag{5.1}$$

where R_t is the area of the target rectangular box and R_g is the area of the ground truth rectangular box. $\cap$ is known as Intersection, $\cup$ is Union

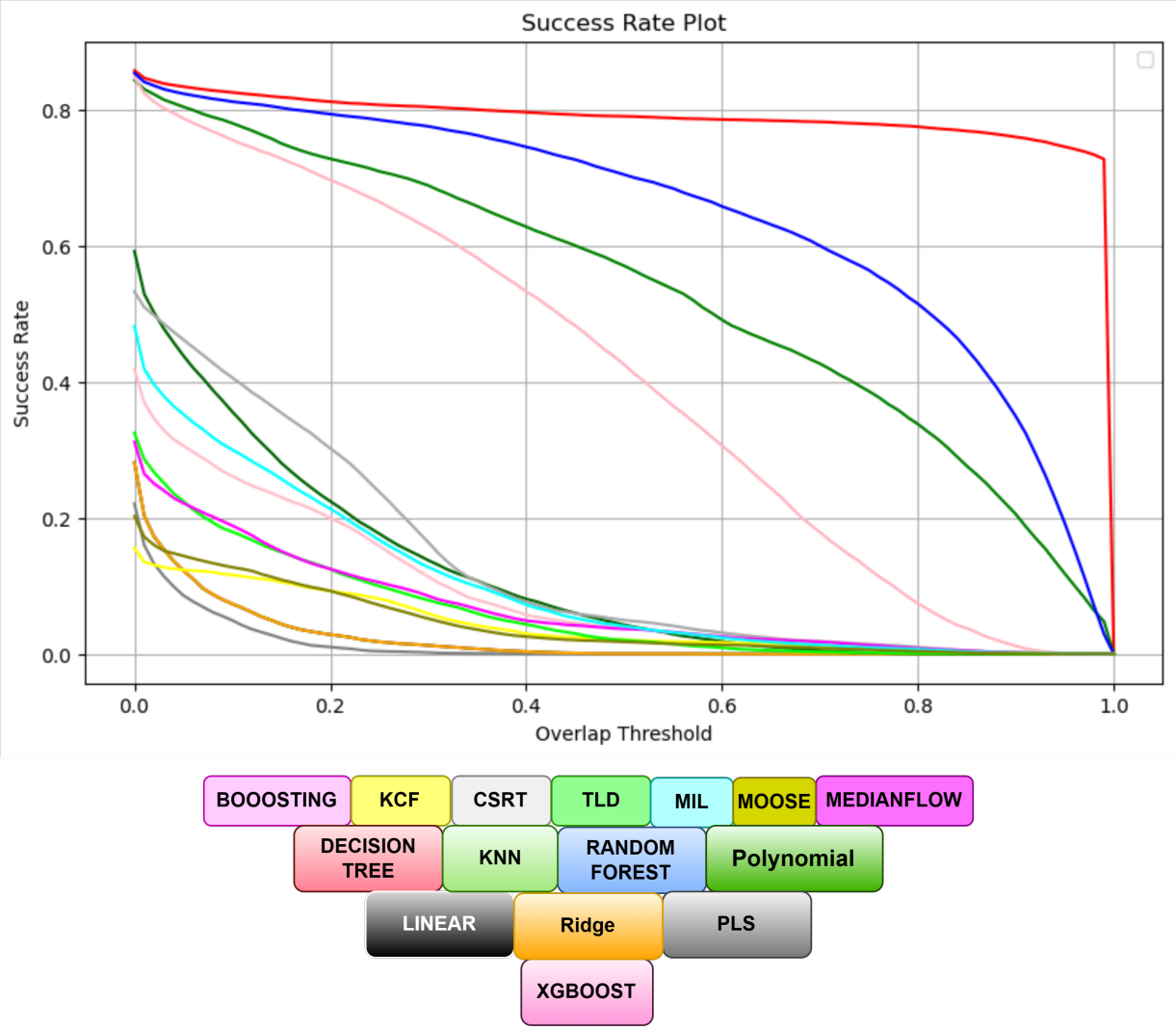


Figure 5.2: A success rate comparison at different IOU thresholds of the proposed tracker and different existing tracker in Visual Object tracking benchmark dataset (VOT2015) [27]. The different color indicates the different tracker as shown above

Success is defined when the IoU surpasses a predefined threshold. To evaluate the tracker's performance, the threshold is varied within the range [0, 1], and the success rate is calculated accordingly.

$$Success\ Rate = \frac{TP}{TP + FP + FN} \tag{5.2}$$

where, TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative.

The success rate plot is plotted against the threshold values vs success rate. A higher success rate on the plot indicates better tracker performance. This emphasizes the robustness and effectiveness of the proposed tracker. Figure 5.2 shows the success plots of different existing models and the proposed model using various regression models. By presenting these success plots, it becomes evident that the proposed tracker achieves superior results compared to alternative models. This evaluation approach, based on varying IoU thresholds and success rates, provides a comprehensive understanding of the tracker's performance across diverse scenarios. The Figure 5.2 shows that the success rate of meta-learner using Decision Tree, Random Forest and KNN provide better results than other models in VOT 2015 benchmark dataset. On the other hand, Figure 5.3, 5.5, and 5.5 show success plots of proposed model with different video sequences of the OTB50 dataset.

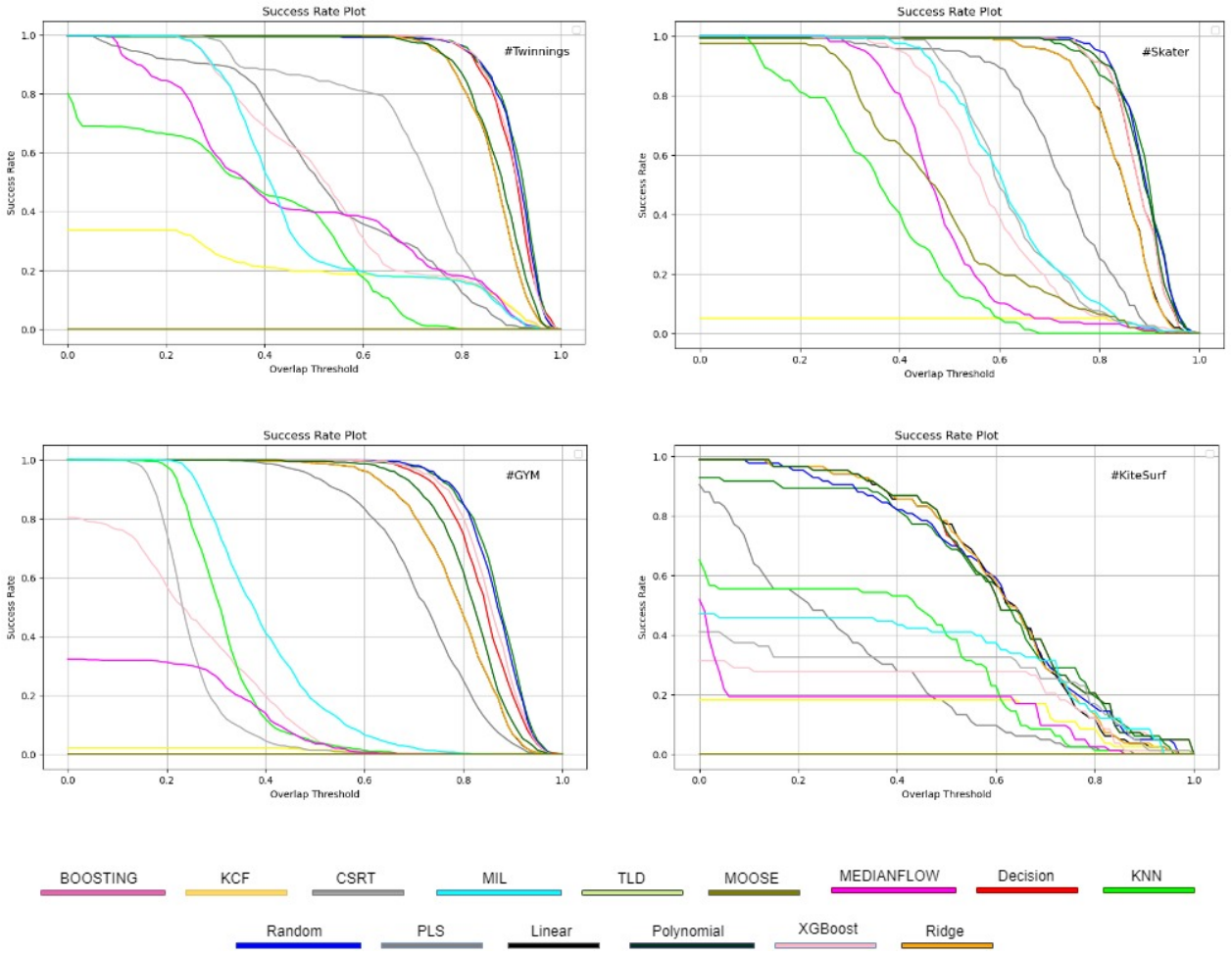


Figure 5.3: Comparison of different tracker using Success plot in OTB 50 Dataset [47].

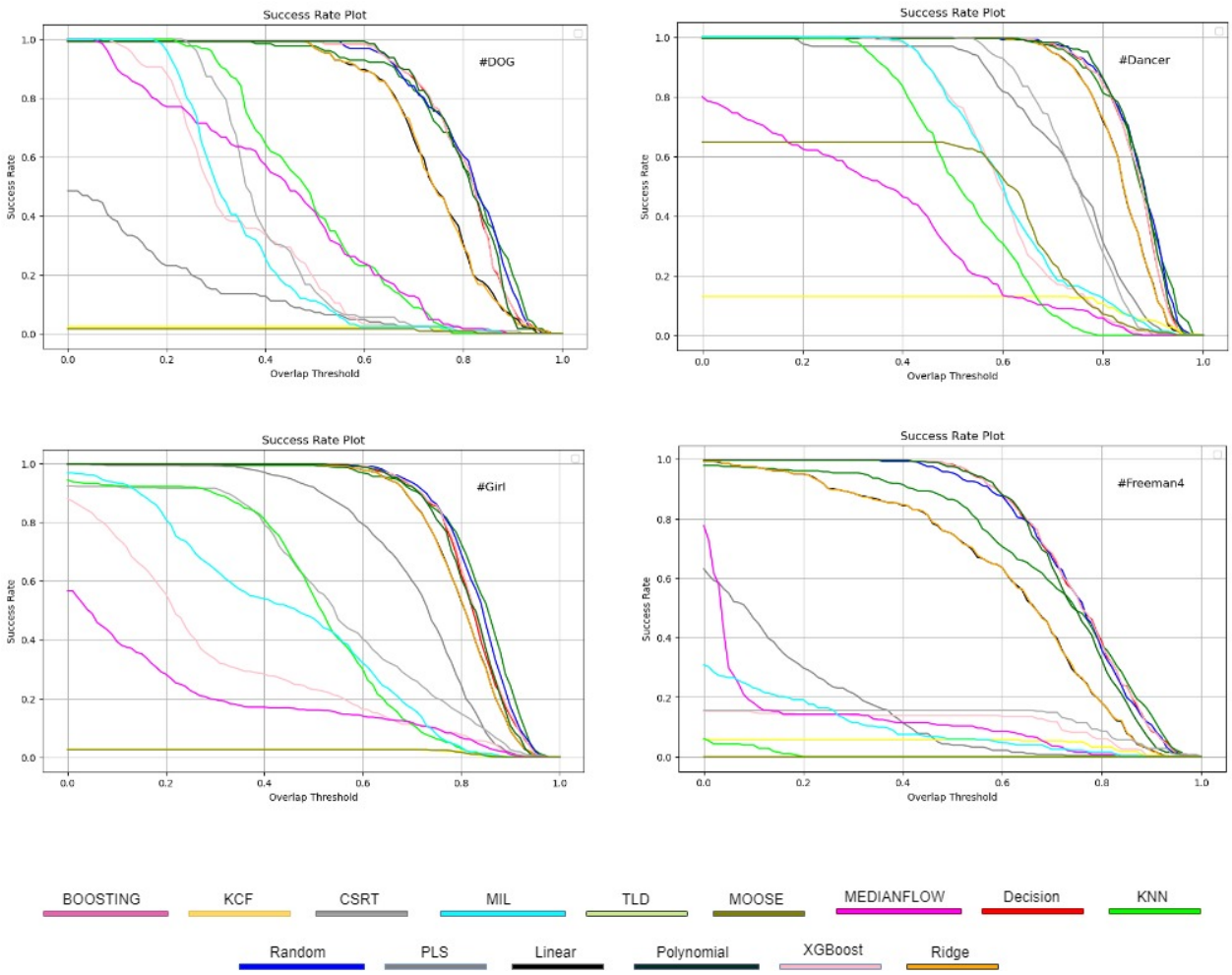


Figure 5.4: Comparison of different tracker using Success plot in OTB 50 Dataset [47].

5.1.2 Center Location Error Plot :

The center location error is basically the Euclidean distance between the centroids of the ground truth and the corresponding tracking result. Center location error is given by

$$Center\ location\ error = \sqrt{(x_g - x_r)^2 + (y_g - y_r)^2} \tag{5.3}$$

The center location error plot is generated by plotting the center location error of each frame against the frame number. The Figure 5.6 shows that the meta-learner using Decision Tree, Random Forest, KNN and XGBOOST gives minimum error than other models in VOT 2015 benchmark dataset. On the other hand, Figure 5.7, 5.8, and 5.9 show success plots of proposed model with different video sequences of the OTB50 dataset.

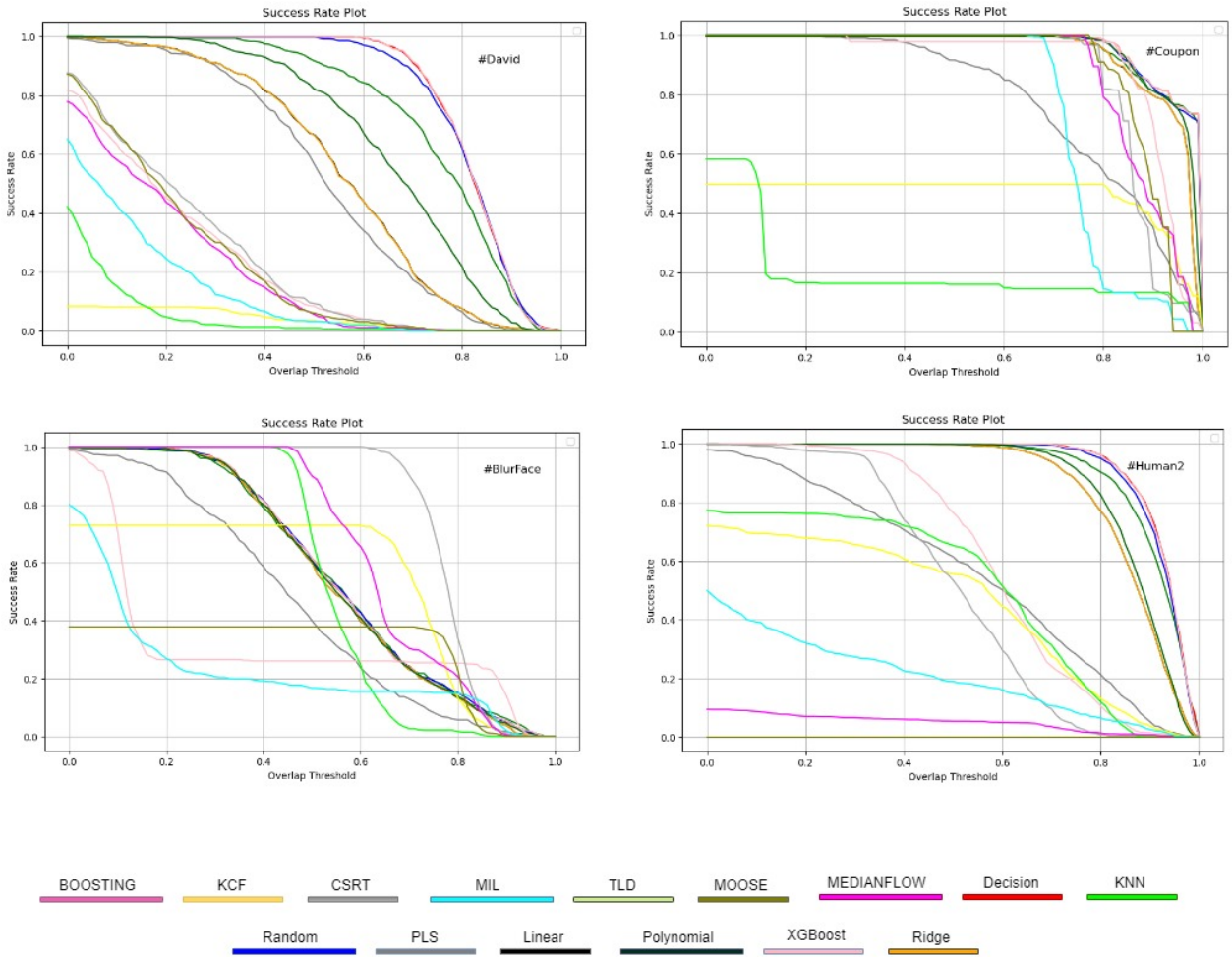


Figure 5.5: Comparison of different tracker using Success plot in OTB 50 Dataset [47].

5.1.3 Precision Plot :

The Precision Plot is another widely used metric for measuring how often a tracker model tracks correctly. It indicates the accuracy of the detected objects and how many detections are correct. This plot is useful for observing the tracker performance but when tracker performance cannot be tracked, it simply looks at the center location error plot. A center location error plot is used to calculate the precision plot. In any frame if the center location error is greater than the given threshold value, then it is considered a negative frame; else, it is considered a positive frame. Precision is given by

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{5.4}$$

To create a Precision Plot, I calculate the precision for various threshold values, ranging from 0 to 50. Thresholds above 50 are not considered, as they lead to excessive error deviation in the center location, making those values less useful. In the OTB Dataset, I have employed a threshold value of 20. The Precision Plot helps me analyze how well the tracker

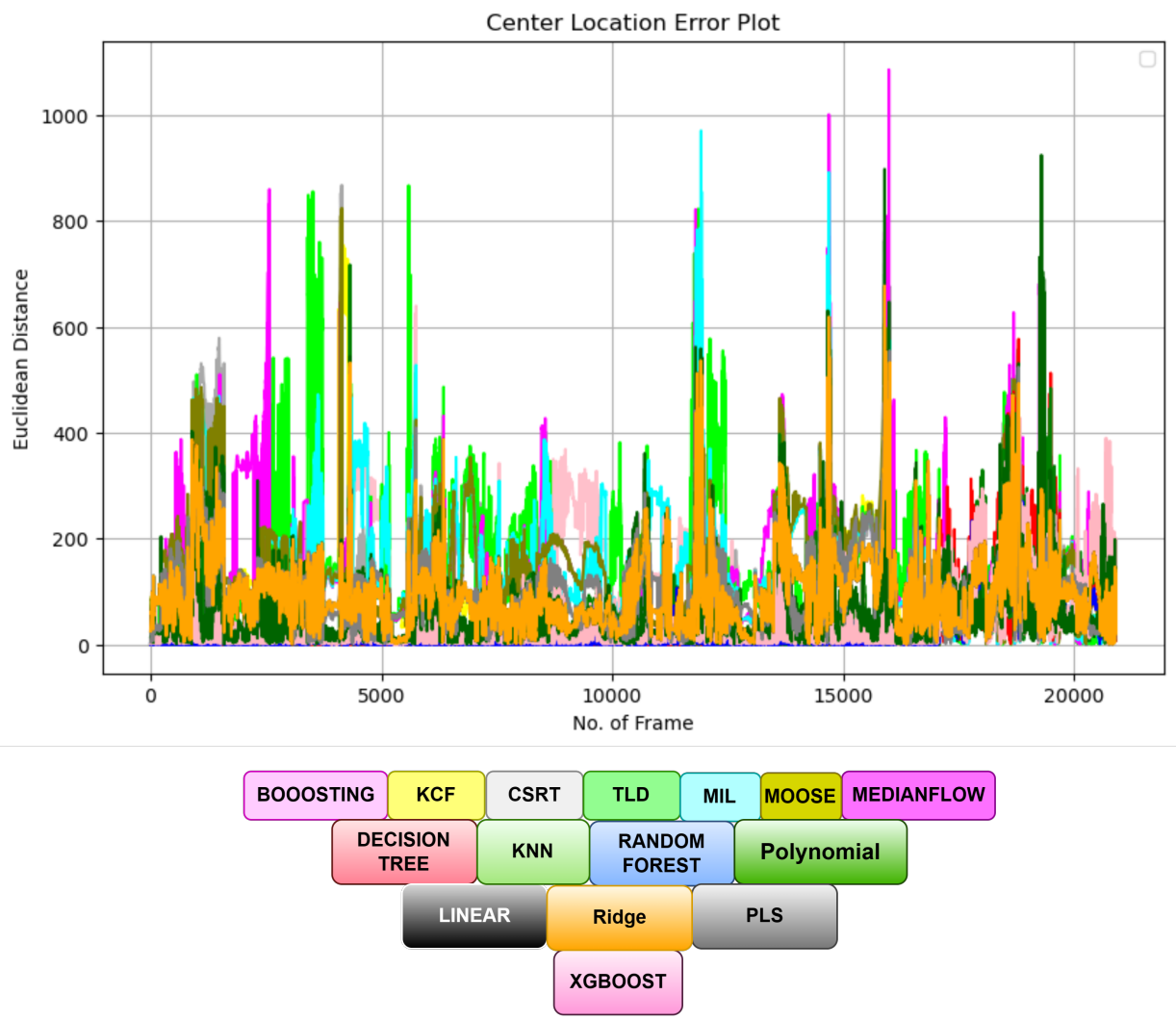


Figure 5.6: A center location error (Euclidean distance) comparison of the proposed tracker and different existing tracker in Visual Object tracking benchmark dataset (VOT2015) [27]. The different color indicates the different tracker as shown above

performs in different scenarios. Figure 5.10 shows that the precision value of meta-learner using Decision Tree, Random Forest, XGBOOST and KNN gives better results than other models in VOT 2015 benchmark dataset. On the other hand, Figure 5.11, 5.12, and 5.13 show success plots of proposed model with different video sequences of the OTB50 dataset.

5.1.4 Tracking results :

Object with Blur and Non-rigid object deformation:

I have tested the proposed model's robustness against blur effects using video sequences like BlurCar3, BlurCar4, BlurFace, BlurOwl, Board, Box, Boy, ClifBar, David, Deer, Drag-



Figure 5.7: Comparison of different tracker using Center location error in OTB 50 Dataset [47].

onBaby, FleetFace, Girl2, Human2, Human7, Human9, Ironman, Jump, Jumping, Liquor, MotorRolling, Soccer, Tiger1, Tiger2, and Woman. The CSRT tracker can handle blur to a certain extent, but it may struggle with detection and require more time to calculate the object's location when the frame is very blurry.

For non-rigid object deformation, I have used video sequences such as Basketball, Bird1, Bird2, BlurBody, Bolt, Bolt2, Couple, Crossing, Crowds, Dancer, Dancer2, David, David3, Diving, Dog, Dudek, FleetFace, Girl2, Gym, Human3, Human4.2, Human5, Human6, Human7, Human8, Human9, Jogging.1, Jogging.2, Jump, Mhyang, Panda, Singer2, Skater, Skater2, Skating1, Skating2.1, Skating2.2, Skiing, Subway, Tiger1, Tiger2, Trans, and Walking. The CSRT tracker excels at handling shape changes in objects due to several key features. It uses a specialized filter that considers various parts and areas of the object, making it adaptable to shape and appearance changes. CSRT also combines features like color and gradient to better distinguish the object from the background, resulting in improved tracking accuracy. The tracker continually updates itself to keep up with the object's changing appearance, and it employs spatial regularization to maintain the object's correct position

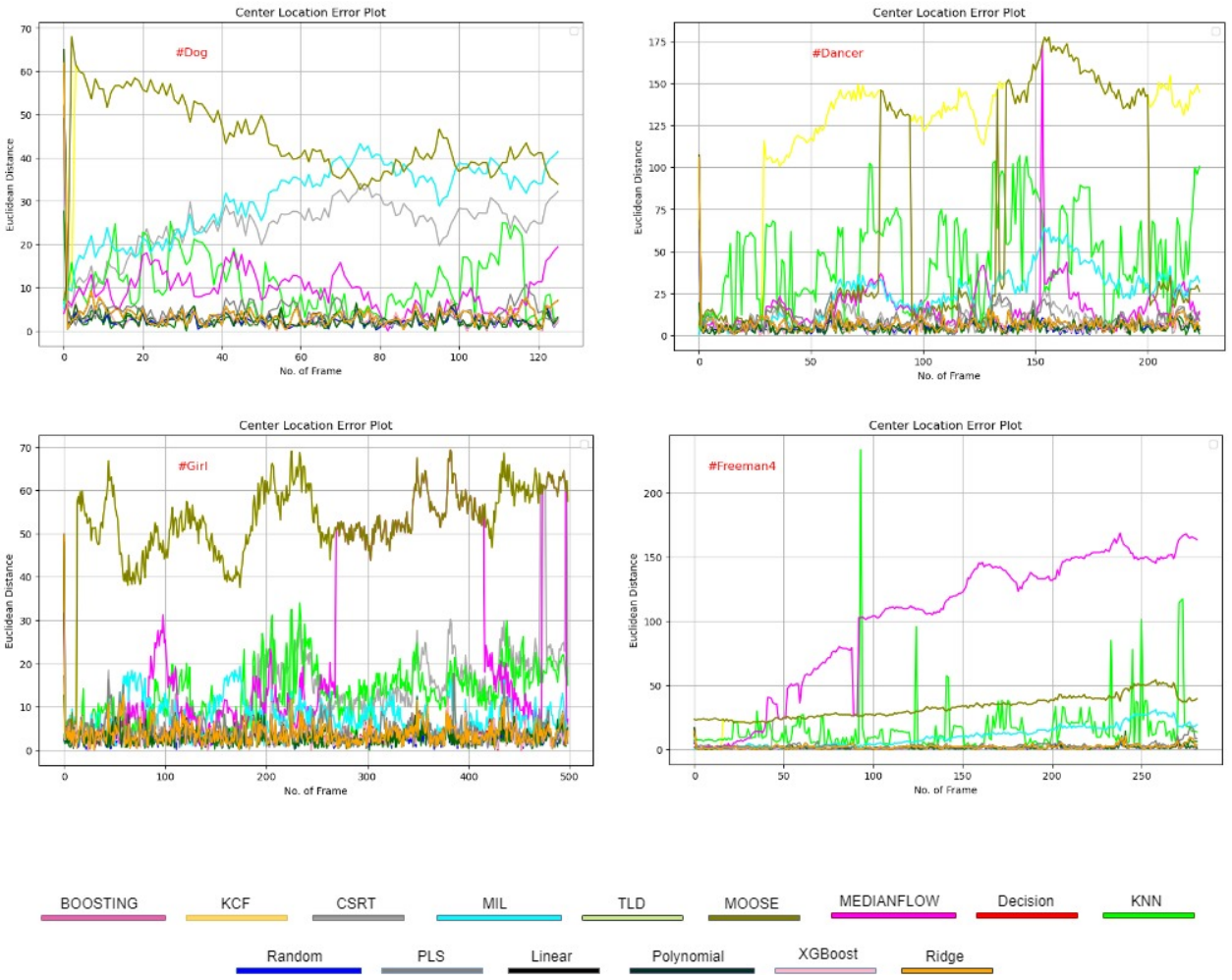


Figure 5.8: Comparison of different tracker using Center location error in OTB 50 Dataset [47].

even when it deforms.

However, some popular tracking methods like MIL, TLD, Boosting, MOOSE, KCF, and MedianFlow have limitations when it comes to shape-changing objects. MIL struggles with significant shape changes due to its use of multiple instances, while TLD, although good at long-term tracking and handling obstacles, is not very adaptable to shape changes. Boosting is not an ideal choice for tracking rapid shape changes, and MOOSE is primarily designed for motion estimation, making it less suitable for handling changes in object appearance. KCF, despite its efficiency, lacks the advanced feature integration and spatial regularization found in CSRT, which makes it less effective for tracking shape-changing objects. MedianFlow is great for tracking small and smooth movements, but it may not perform well with significant deformations or rapid changes. By using a combination of various trackers, My model achieves excellent tracking results, particularly in cases of blurriness and appearance changes.

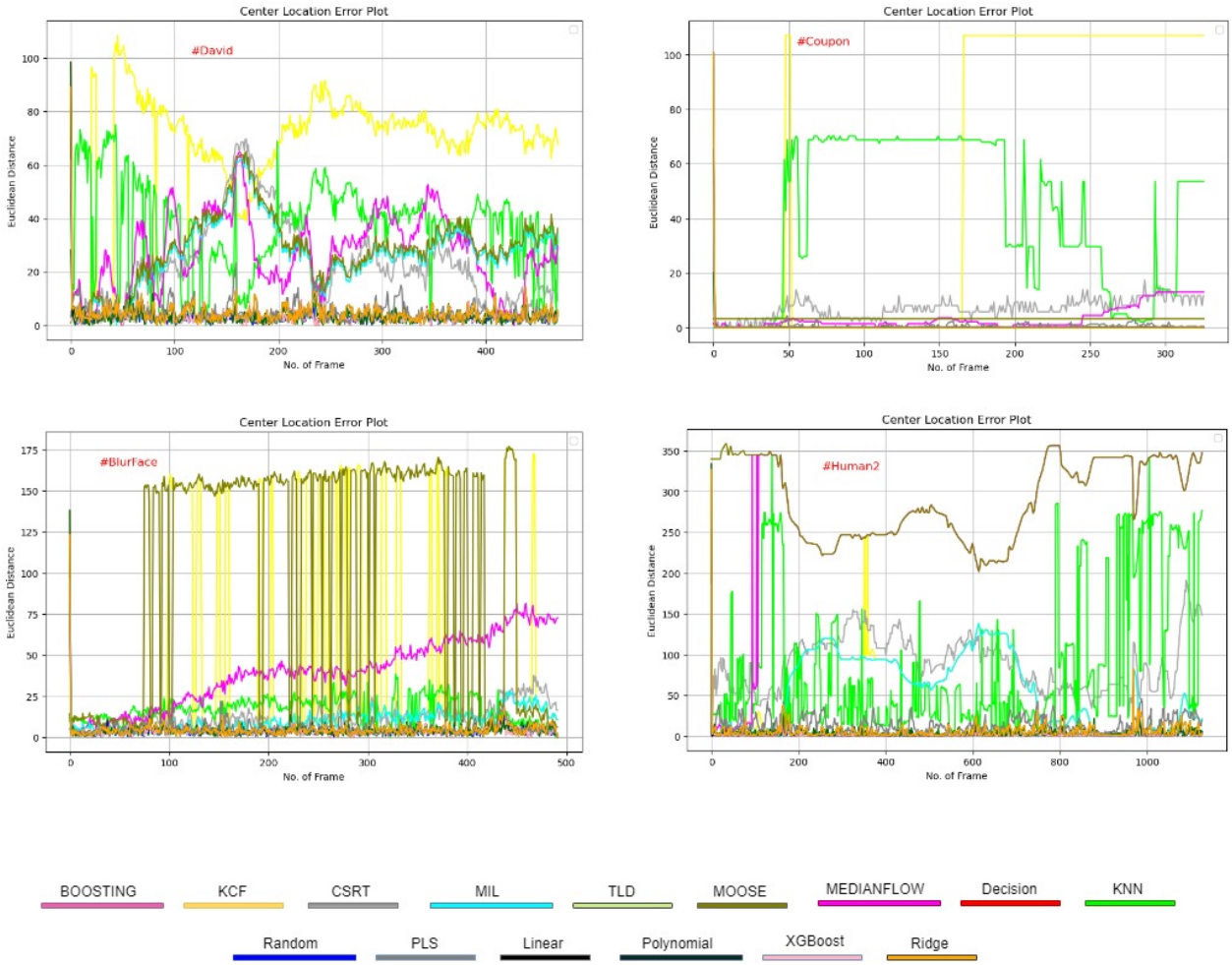


Figure 5.9: Comparison of different tracker using Center location error in OTB 50 Dataset [47].

Nonlinear Motion and high velocity :

To evaluate the proposed model's ability to handle non-linear and fast motion, I have used video sequences such as Car4, BlurBody, Deer, DragonBaby, Jumping, and Boy. I have compared the tracking results with those of some state-of-the-art trackers. In these videos, the target objects exhibit non-linear motion with high velocity.

In the Car24 video sequence, the object's movement is initially linear and slow, which enables the CSRT, Boosting, and MIL trackers to perform effectively. However, after the 25th frame, as the object accelerates, the MIL tracker fails to keep up with the target. The Boosting tracker manages to track the object well due to its linear motion, while the CSRT, KCF, and MedianFlow trackers deliver the most accurate results overall.

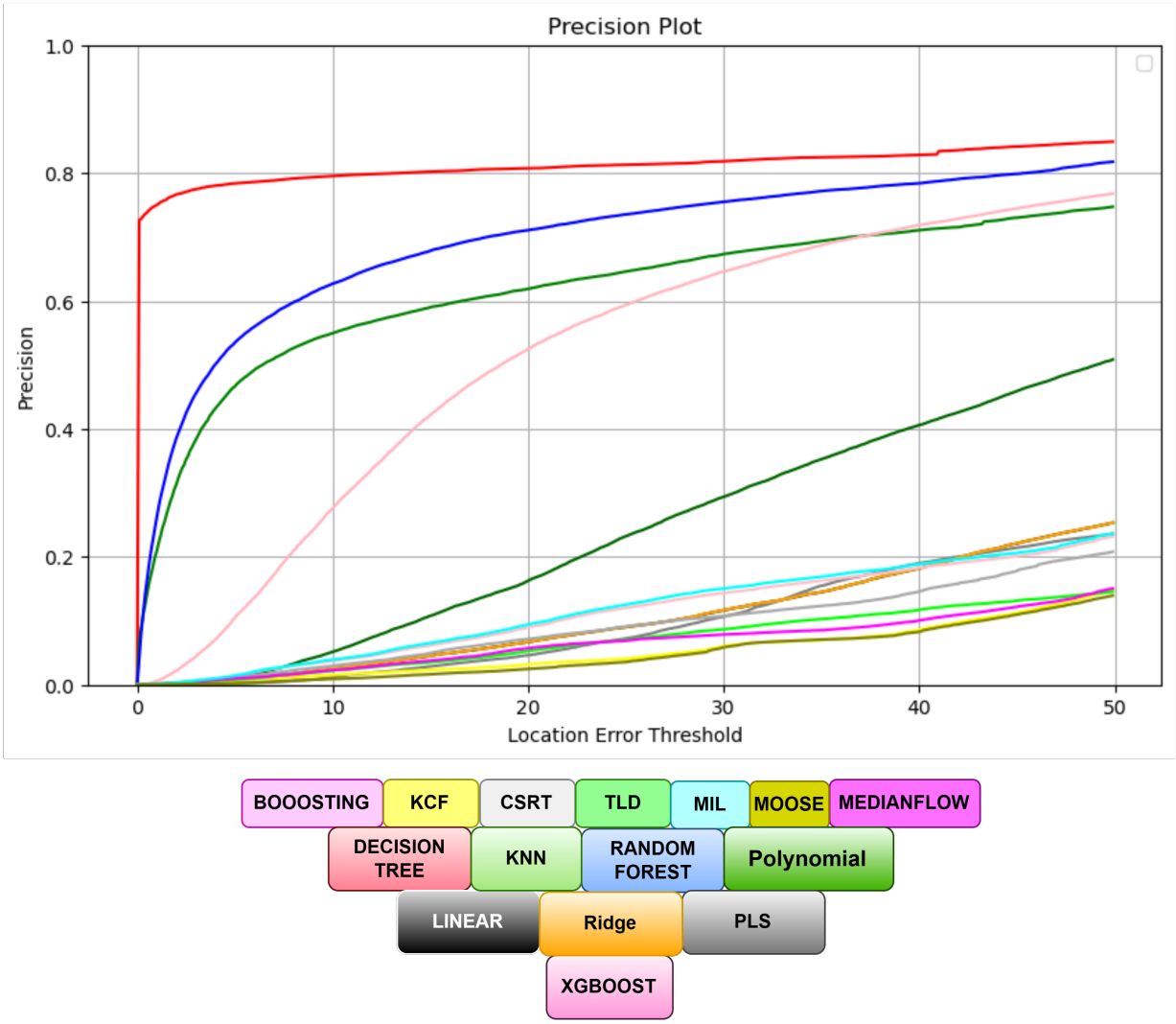


Figure 5.10: A precision at different center location threshold comparison of the proposed tracker and different existing tracker in Visual Object tracking benchmark dataset (VOT2015) [27]. The different color indicates the different tracker as shown above

Partially or fully occluded Object Tracking :

Occlusion, whether it's full or partial, is a major challenge in object tracking, as it results in the temporary loss of visual information about the target. The effectiveness of different tracking algorithms under occlusion can vary significantly. To demonstrate the improved performance of My proposed model against occluded objects, I have used the following video sequences: DragonBaby, Jogging, Rubik, David3, FaceOcc1, Walking, Walking2, and Jogging2. For instance, in Jogging1 and Jogging2, the objects are fully occluded by an electric pole. The TLD tracker excels in this scenario due to its integrated detection mechanism, which enables the re-detection and re-initialization of the target after occlusion. The CSRT tracker is also impressive for its ability to handle partial occlusions by concentrating on the most reliable parts of the target's appearance. On the other hand, trackers like Boosting,

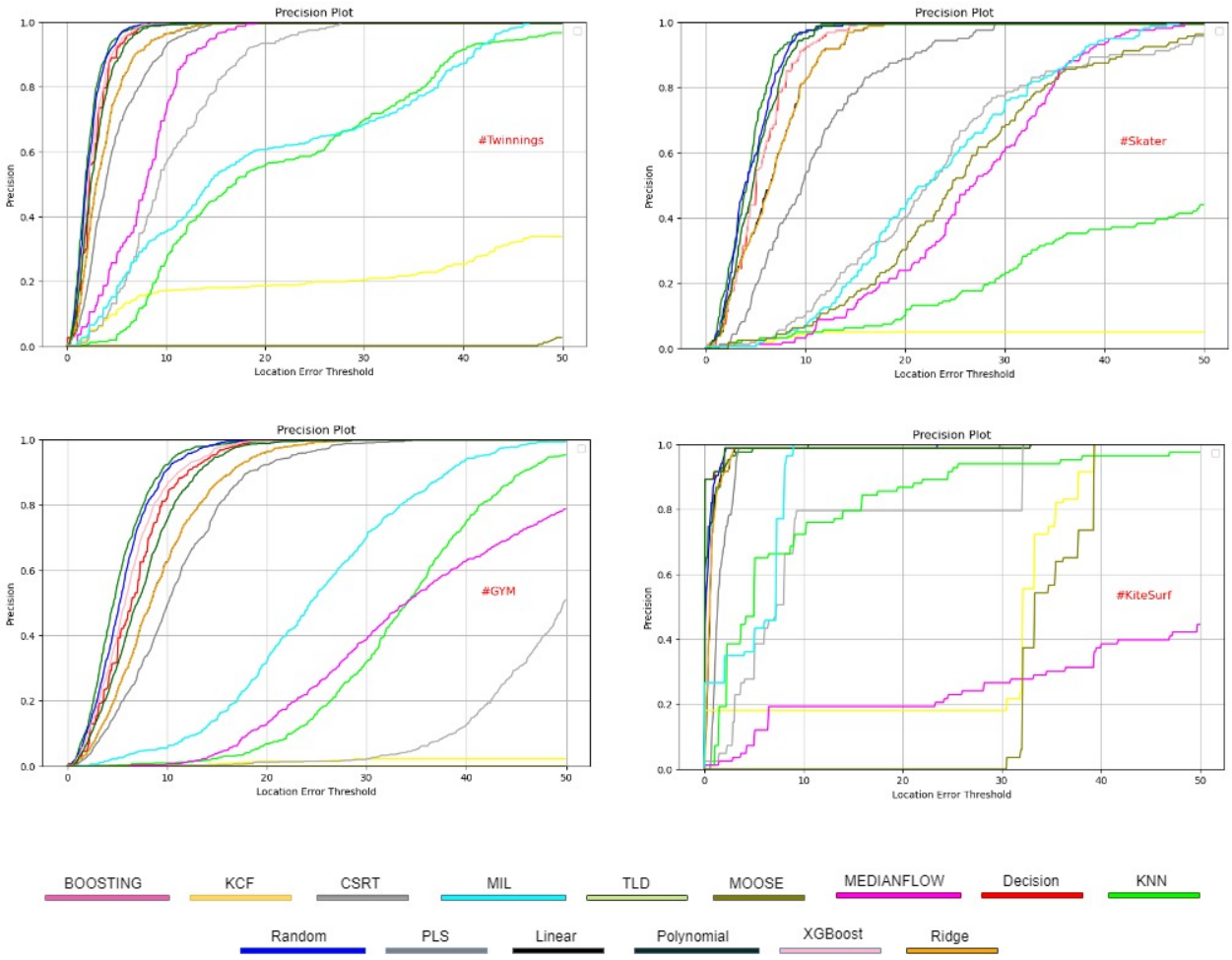


Figure 5.11: Comparison of different tracker using Precision Plot in OTB 50 Dataset [47]

MIL, KCF, MedianFlow, and MOSSE have limitations in effectively managing occlusions, especially full occlusions.

Low Resolution :

Tracking objects in low-resolution videos can be quite challenging, as the lack of fine details and texture information makes it difficult for many tracking algorithms to maintain accuracy. In such situations, color histograms, which represent the color distribution within the target region, can be particularly useful due to their resilience against scale, rotation, and partial occlusion changes.

When it comes to low-resolution tracking scenarios that utilize color histogram features, the CSRT tracker is considered the most robust option. Its ability to focus on the most reliable parts of the target's appearance, combined with the inherent robustness of discriminative correlation filters, makes it well-equipped to handle the challenges posed by low-resolution inputs. The KCF tracker is also a decent choice, as it benefits from color histograms to maintain tracking accuracy to a certain extent.

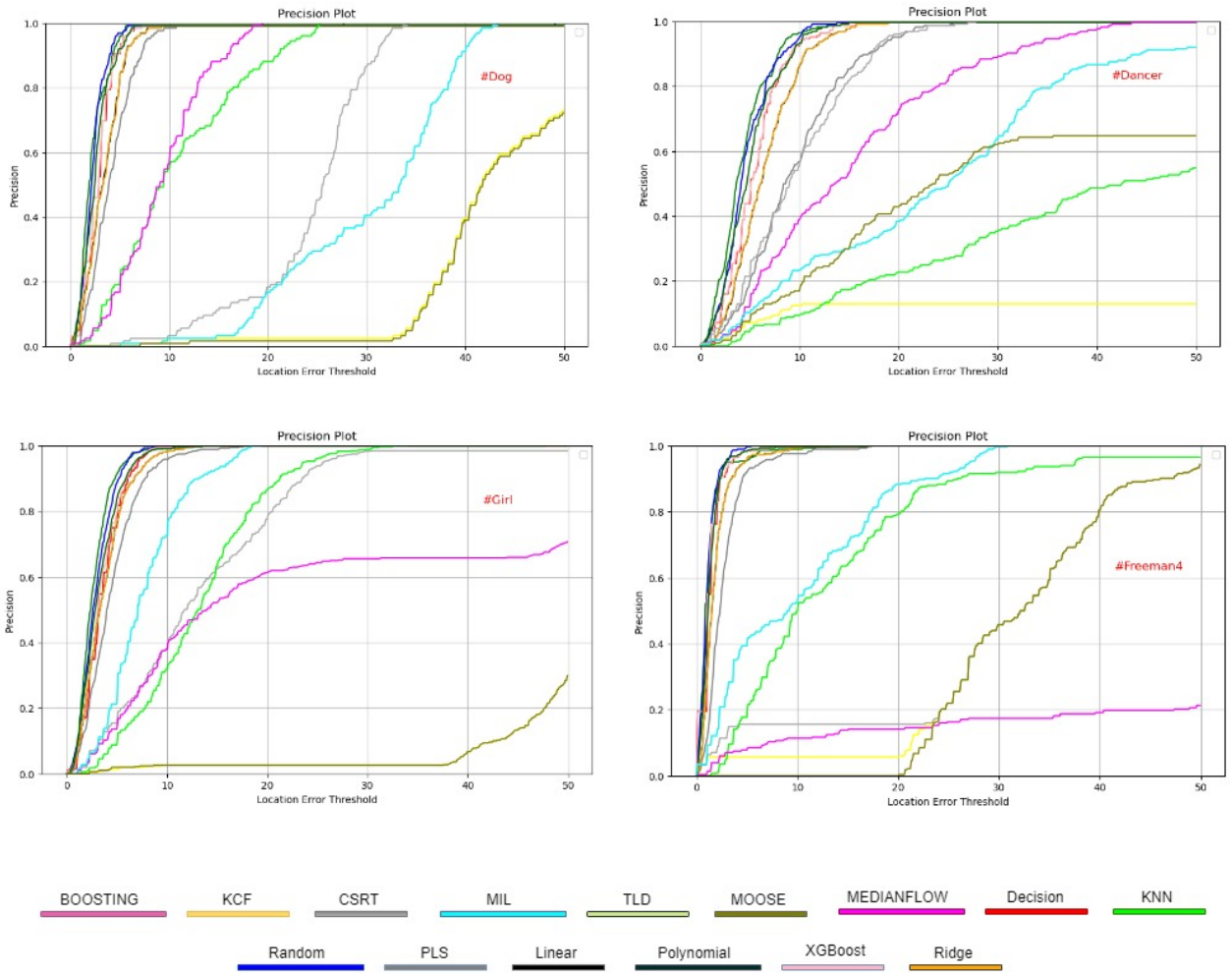


Figure 5.12: Comparison of different tracker using Precision Plot in OTB 50 Dataset [47]

Other trackers, such as MIL, TLD, Boosting, MedianFlow, and MOSSE, have varying levels of performance in low-resolution scenarios, but they generally struggle more compared to CSRT and KCF. Therefore, for tasks involving low-resolution video tracking, it is recommended to use the CSRT tracker with color histogram features for its superior performance.

To evaluate the performance of the tracker against low-resolution videos, I have used the following video sequences: Deer, Walking2, and Ironman. The results obtained from these tests are satisfactory for low-resolution video sequences.

Illumination Variation :

of the target, potentially leading to tracking failures. Different trackers have varying levels of success in handling illumination changes, depending on their underlying algorithms and the features they utilize.

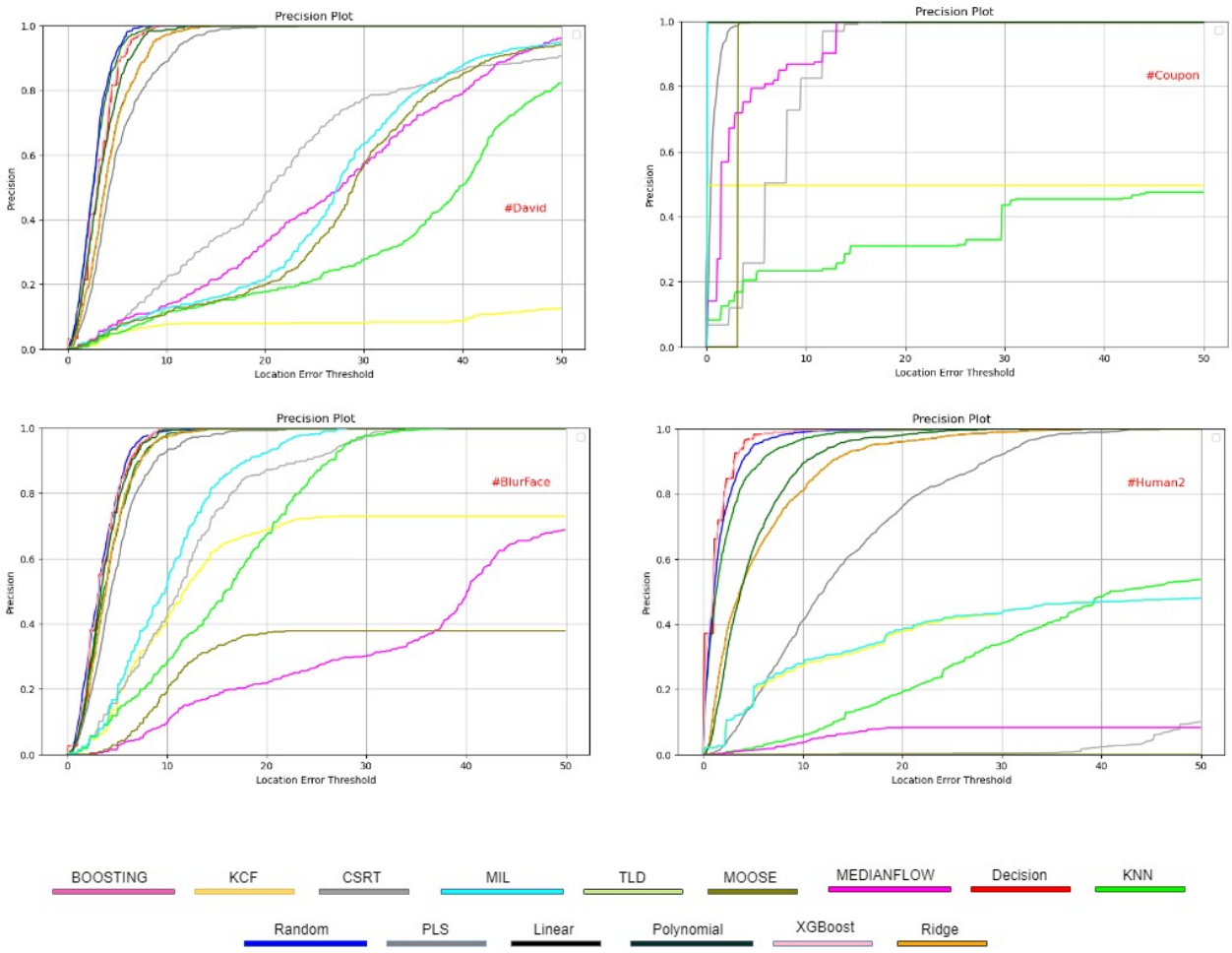


Figure 5.13: Comparison of different tracker using Precision Plot in OTB 50 Dataset [47]

In cases of illumination variation, the CSRT tracker is considered the most robust option. This is due to its advanced feature representation, which includes channel and spatial reliability. CSRT's ability to adapt to changes in lighting by focusing on stable target features makes it highly effective. The tracker also weighs the reliability of different parts of the target's appearance, which helps it maintain robust tracking even under significant illumination variations.

Other trackers, such as MIL, TLD, KCF, and MOSSE, have moderate performance but can struggle with more severe changes in lighting. Boosting and MedianFlow trackers are less effective under illumination variations, as they rely on consistent appearance models and fine motion details, respectively. To assess the effectiveness of the proposed model against illumination changes, I have used the video sequences KiteSurf, Skating1, Man, Crowds, and Singer1. The results obtained from these tests are quite impressive, demonstrating the model's ability to handle varying lighting conditions.

Background Cluster :

Tracking objects in the presence of background clutter is challenging due to the increased likelihood of confusion between the target and the cluttered background. Color histogram-based tracker models have problems tracking this type of object. To address this issue, multi-scale template matching can be used, allowing the tracker to adapt to changes in the target's size and appearance, which improves robustness.

In scenarios with a lot of background clutter, the TLD tracker combined with multi-scale template matching produces the best results. Its integrated approach of tracking, learning, and detection, combined with the ability to match targets across various scales, ensures resilience to background clutter. Although The CSRT tracker also performs well due to its advanced feature representation and reliability measures, but it lacks explicit multi-scale template matching. Other trackers, such as MIL, KCF, Boosting, MedianFlow, and MOSSE, are less effective in handling background clutter, especially without multi-scale template matching capabilities. The Deer, David3, Car24, and Crowds video sequences are used to evaluate the tracker performance, which is shown in Figure 5.14.

In-plane Rotation :

In-plane rotation refers to when the target object rotates within the image frame. To demonstrate the effectiveness of the new tracker model against in-plane rotation, the video sequences BlurBody, Dancer, DragonBaby, Deer, Ironman, Rubik, Diving, and Boy are used. The performance comparison of the proposed tracker is shown in Figure 5.15.

Scale variation :

When an object moves towards or away from the camera, its size can change significantly. To address this issue, multi-scale template matching is implemented, which yields good results. The Dog1, Walking2, Twinings, Boy, Car24, Dog, Diving, Walking, Rubik, Ironman, DragonBaby, Dancer, BlurBody, and Car4 video sequences are used to demonstrate the robustness of the tracker against scale variations.

In the Walking2 and Dog video sequences, the target objects move away from the camera, which causes their size to decrease. Traditional trackers like Boosting, MIL, CSRT, KCF, and MedianFlow can efficiently track the object without multi-scale template matching. However, their bounding boxes do not adapt to the size changes of the objects in the scale-space. On the other hand, TLD trackers with multi-scale template matching can track the object while simultaneously adjusting the bounding box size to better fit the target.

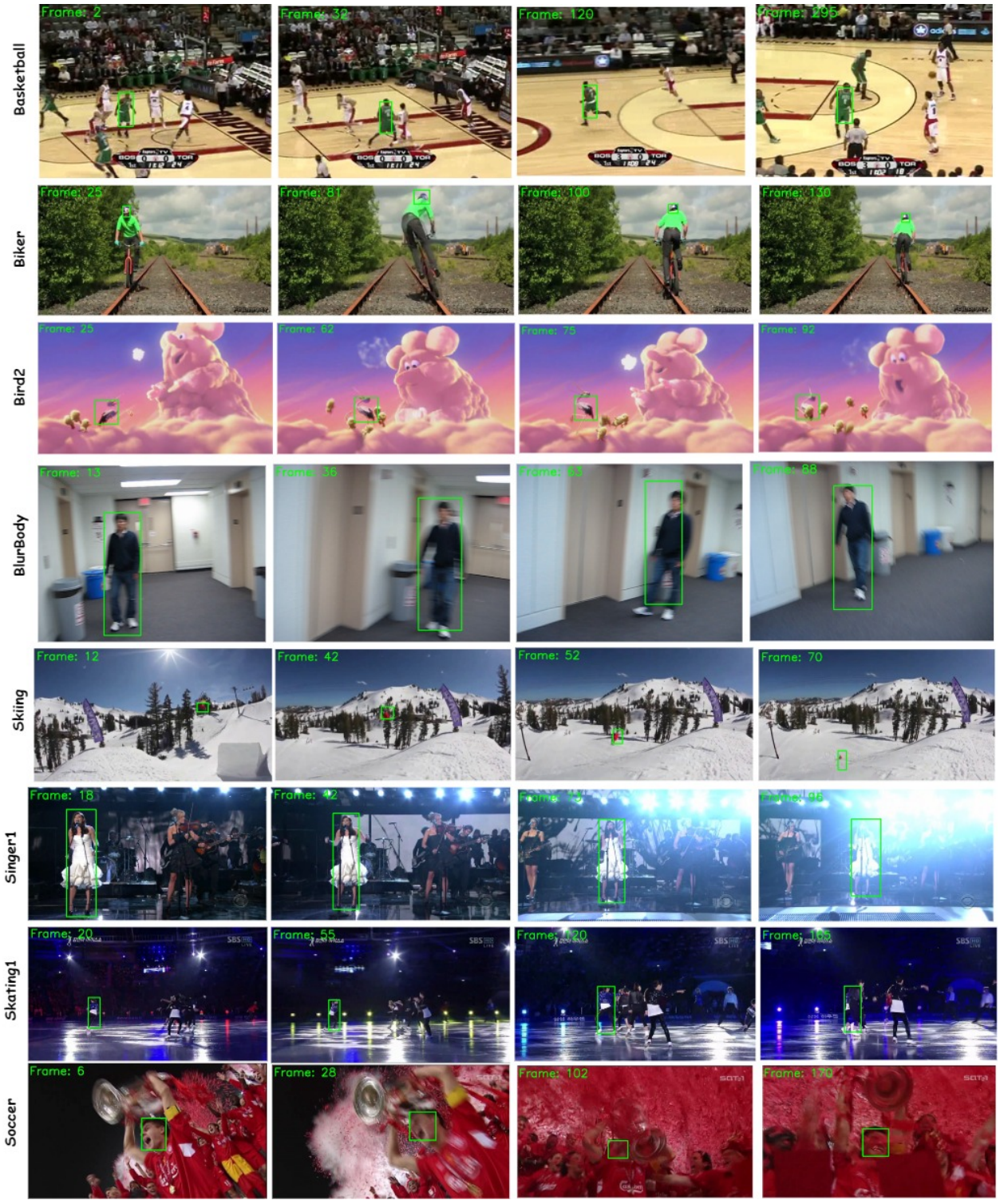


Figure 5.14: Tracking result in various challenging situations

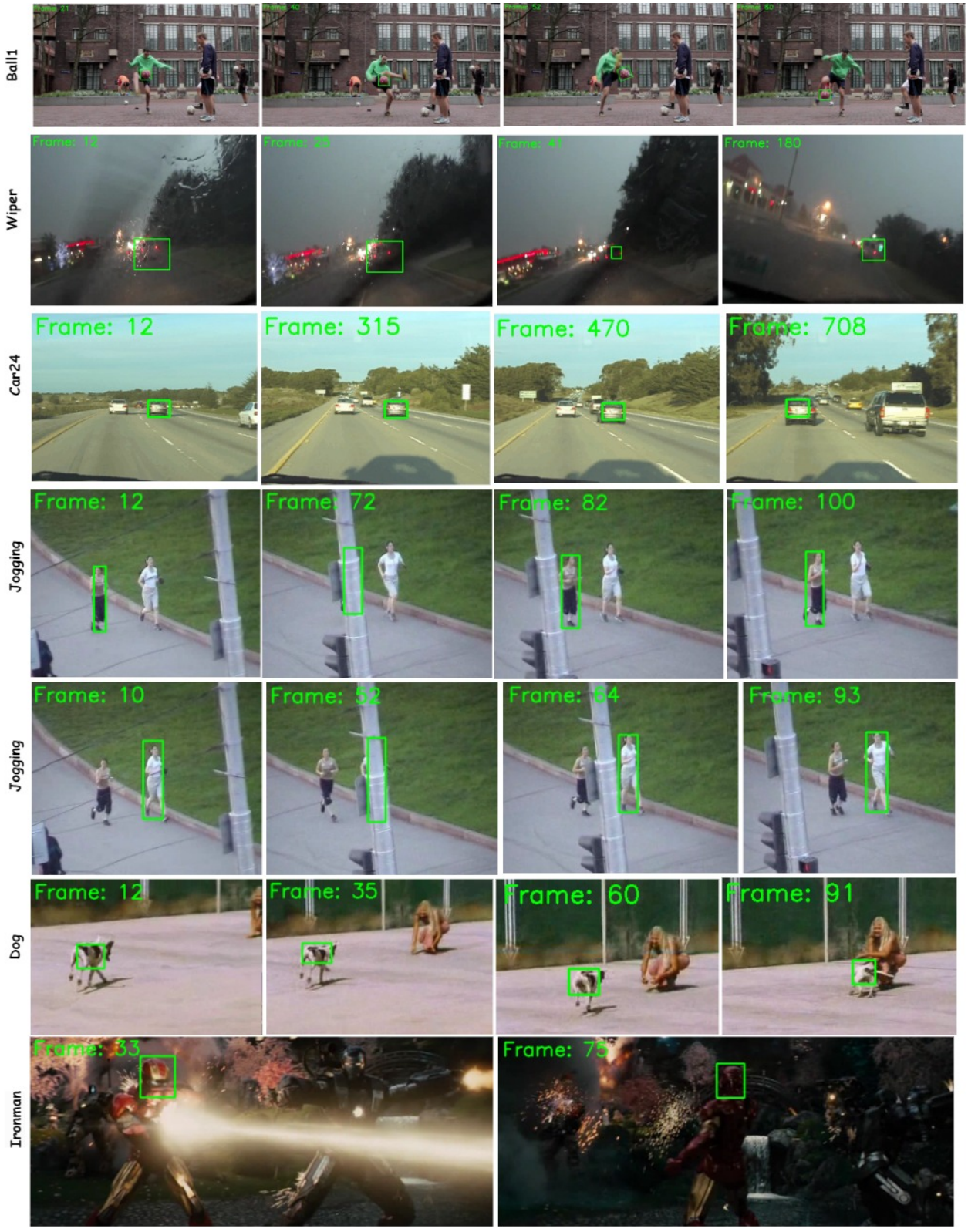


Figure 5.15: Tracking result in various challenging situations

5.2 Quantitative Comparison :

5.2.1 State-of-the-art comparison :

The proposed method is compared with fourteen state-of-the-art tracking models like BOOSTING [17], KCF[21], CSRT [32], TLD [52], MEDIANLOW [14], MIL [13], MOOSE [15], MDNet [38], DeepSRDCF [48], EBT [53], SRDCF [12], LDP [33], sPST [23] and SC-EBT [43]. I used mainly four metrics which are Object Tracking Error, Precision, Expected Average Overlap (EAO), and Success Rate to compare all the trackers shown in Table 5.3, tested on Visual Object Tracking Benchmark.

Table 5.3: Success rate, Precision, Object Tracking Error, Expected Average Overlap and Frame Rate comparison of proposed tracker in Visual Object Tracking Benchmark dataset (VOT2015) [27]

Trackers	**OTE**	**Precision**	**EAO**	**Success Rate**	**Frame Rate**
BOOSTING [17]	142.64	0.23	0.09	0.03	29.15
KCF [21]	126.75	0.14	0.04	0.02	204.74
CSRT [32]	129.80	0.20	0.13	0.04	20.10
TLD [52]	149.84	0.14	0.05	0.02	15.30
MEDIANFLOW [14]	142.89	0.15	0.06	0.03	513.82
MIL [13]	139.10	0.23	0.10	0.04	21.18
MOOSE [15]	133.78	0.14	0.04	0.01	2106.82
MDNet [38]	-	-	0.38	0.60	1.14
DeepSRDCF [48]	-	-	0.31	0.56	2.63
EBT [53]	-	-	0.31	0.47	0.56
SRDCF [12]	-	-	0.29	0.56	0.50
LDP [33]	-	-	0.27	0.51	0.22
sPST [23]	-	-	0.27	0.55	0.99
SC-EBT [43]	-	-	0.25	0.55	1.25
LINEAR	95.43	0.25	0.02	0.00	15.80
RIDGE	95.42	0.25	0.02	0.00	15.47
Decision Tree	22.35	0.84	0.79	0.79	15.97
Random Forest	26.35	0.81	0.63	0.74	13.47
XGBOOST	40.97	0.76	0.40	0.53	12.16
KNN	38.40	0.74	0.52	0.62	12.90
Polynomial	66.96	0.50	0.11	0.08	14.90
PLS	106.95	0.23	0.01	0.00	15.13

5.2.2 Object Tracking Error :

The Object Tracking Error (OTE) is the average of the Euclidean distance between the actual and estimated positions of the item throughout the entire video sequence.

$$OTE = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_g - x_r)^2 + (y_g - y_r)^2} \tag{5.5}$$

where (x_g^i, y_g^i) = coordinate of the centroid of ground truth for i_{th} frame, (x_r^i, y_r^i) = coordinates of the centroid of the tracking system result for i_{th} frame, N is the number of frame. A lower object tracking error value indicates a better performance of the model. The results shown in Table 5.3 indicate that four of My proposed meta-learner models using Decision Tree, Random Forest, KNN and XGBOOST give an outstanding outcome in terms of Object Tracking Error, achieving the bare minimum Object Tracking Error of 22.35, 26.35, 38.40 and 40.97 respectively in VOT 2015 dataset. Table 5.4 and table 5.5 shows the object tracking error comparison of proposed model in different video sequences of OTB 50 dataset.

TRACKER	Skater	GYM	KiteSurf	Dog	Dancer
BOOSTING	23	24	4	30	24
KCF	137	131	27	43	121
CSRT	23	50	11	24	9
MIL	53	24	4	30	24
TLD	26	34	9	10	45
MEDIAN FLOW	23	41	46	9	15
MOSSE	27	133	34	44	63
DECISION TREE	6	6	0	3	5
RANDOM FOREST	4	5	0	2	4
LINEAR	7	9	1	3	6
RIDGE	7	9	1	3	6
XGBOOST	6	6	0	3	5
KNN	4	5	0	2	4
POLYNOMIAL	5	7	0	3	5
PLS	11	10	2	4	9

Table 5.4: Object tracking error comparison different tracking algorithms

TRACKER	Twinnings	Girl	Freeman4	David	Coupon
BOOSTING	19	7	9.7	27	0.1
KCF	58	52	30	41	53
CSRT	10	13	28	14	6.9
MIL	19	7	9	16	0.2
TLD	21	13	14	22	41
MEDIAN FLOW	7	25	96	16	3.4
MOSSE	76	52	31	17	3.1
DECISION TREE	2	3	1.3	3	0.3
RANDOM FOREST	2	2	1.1	2	0.2
LINEAR	3	3	1.9	4	0.3
RIDGE	4	3	1.9	4	0.3
XGBOOST	2	3	1.3	3	0.3
KNN	2	2	1.3	2	0.0
POLYNOMIAL	3	3	1.3	3	0.3
PLS	4	4	2.7	5	0.9

Table 5.5: Object tracking error comparison different tracking algorithms

5.2.3 Precision :

Precision is nothing but the ratio of true positive frames and total frames, where the frame is called true positive if the center location error is lower than the threshold. The precision is represented by a precision plot that is generated based on a specific threshold for location error. A higher precision value indicates a better performance of the model. As shown in this Table 5.3, Four of My proposed meta-learning models such as Decision Tree, Random Forest, XGBOOST and KNN produce extraordinary outcomes in terms of precision score, reaching the excellence with a spectacular score of 0.84, 0.81, 0.76 and 0.74 respectively in VOT 2015 dataset. On the other hand, table 5.6 and table 5.7 shows the Precision comparison of proposed model in different video sequences of OTB 50 dataset.

5.2.4 Expected Average Overlap (EAO) :

Expected Average Overlap (EAO) is a vital metric for evaluating the accuracy of object-tracking models. The EAO, is computed by determining the overlap percentage between the actual object's location and the prediction made by the tracking model. This computation is performed for every frame concerning the ground truth and the predicted bounding box. The EAO is then calculated as the mean IoU across all frames in a video sequence. The formula for EAO is as follows:

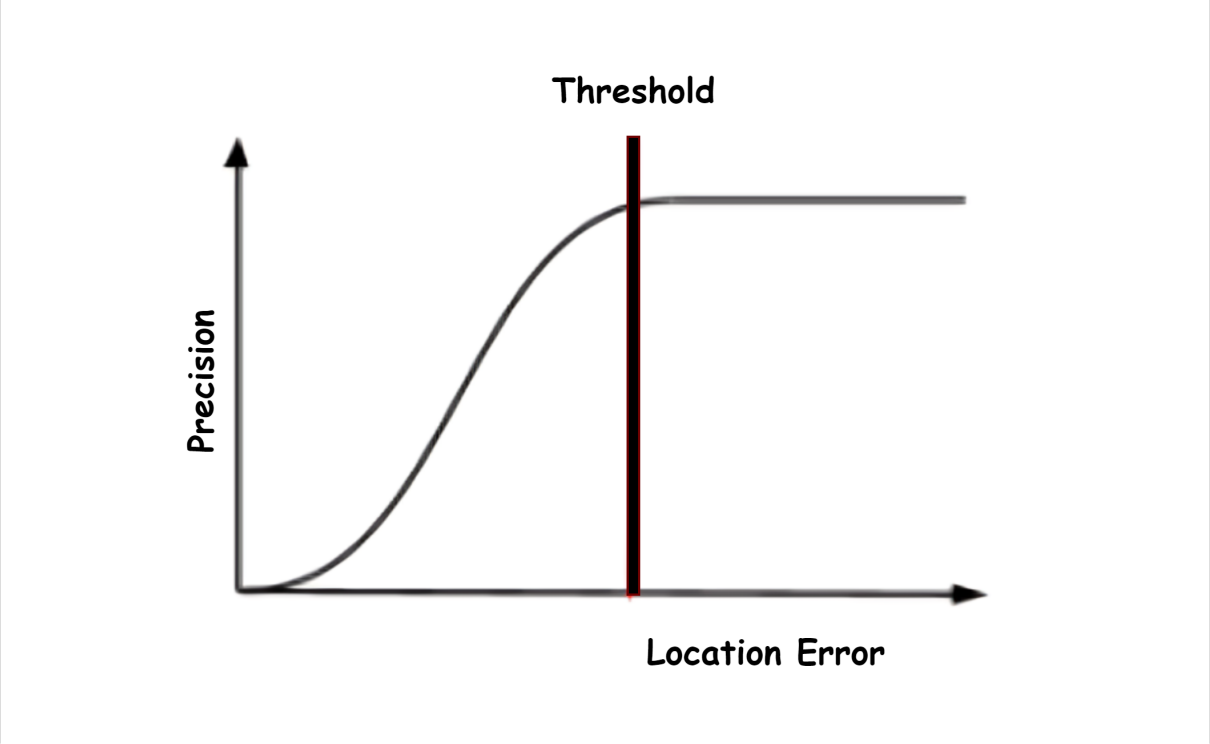


Figure 5.16: An illustration of Precision calculation

$$EAO = \frac{1}{N} \sum_{i=1}^{N} IoU_i \tag{5.6}$$

where IoU_i represents the Intersection Over Union for the $i-th$ frame.

This metric provides an overall assessment of the model's tracking accuracy by averaging the IoU values over the entire video sequence. A higher EAO value indicates the accurate tracking of the model. In this examination, My proposed models, meta-learner using Decision Tree, Random Forest, KNN and XGBOOST, performed superior to the deep learning tracker. These four models demonstrated remarkable tracking accuracy, with an EAO score of 0.79, 0.63, 0.52 and 0.40 respectively in VOT 2015 dataset.And the table 5.8 and table 5.9 shows the expected average overlap comparison of proposed model in different video sequences of OTB 50 dataset.

TRACKER	Skater	GYM	KiteSurf	Dog	Dancer
BOOSTING	0.42	0.31	1.0	0.16	0.38
KCF	0.05	0.01	0.1	0.02	0.12
CSRT	0.40	0.01	0.7	0.18	0.95
MIL	0.42	0.31	1.0	0.16	0.38
TLD	0.10	0.06	0.8	0.88	0.22
MEDIAN FLOW	0.23	0.12	0.1	1.0	0.72
MOSSE	0.30	0.0	0.0	0.01	0.42
DECISION TREE	0.99	0.99	0.98	0.99	0.99
RANDOM FOREST	0.99	0.99	0.98	0.99	0.99
LINEAR	0.99	0.96	0.98	0.99	0.99
RIDGE	0.99	0.96	0.98	0.99	0.99
XGBOOST	0.99	0.99	0.98	0.99	0.99
KNN	0.99	0.99	1.00	0.99	1.0
POLYNOMIAL	0.99	0.98	0.98	0.99	0.99
PLS	0.88	0.92	0.98	0.99	0.95

Table 5.6: Precision comparison of different tracking algorithms at threshold = 20

TRACKER	Twinnings	Girl	Freeman4	David	Coupon
BOOSTING	0.60	1.0	0.8	0.21	0.98
KCF	0.18	0.02	0.0	0.07	0.49
CSRT	0.93	0.77	0.1	0.47	0.98
MIL	0.60	1.0	0.8	0.21	0.98
TLD	0.55	0.86	0.7	0.17	0.30
MEDIAN FLOW	0.97	0.61	0.1	0.32	0.98
MOSSE	0.00	0.02	0.0	0.19	0.97
DECISION TREE	0.99	0.99	1.0	0.99	0.99
RANDOM FOREST	0.99	0.99	1.0	0.99	0.99
LINEAR	0.99	0.99	1.0	0.99	0.99
RIDGE	0.99	0.99	1.0	0.99	0.99
XGBOOST	0.99	0.99	1.0	0.99	0.99
KNN	0.99	1.0	1.0	0.99	0.99
POLYNOMIAL	0.99	0.99	1.0	0.99	0.99
PLS	0.99	1.0	1.0	0.99	0.99

Table 5.7: Precision comparison of different tracking algorithms at threshold = 20

TRACKER	Skater	GYM	KiteSurf	Dog	Dancer
BOOSTING	0.58	0.23	0.21	0.33	0.60
KCF	0.04	0.01	0.14	0.01	0.11
CSRT	0.62	0.24	0.26	0.39	0.73
MIL	0.61	0.39	0.33	0.33	0.61
TLD	0.35	0.31	0.31	0.48	0.52
MEDIAN FLOW	0.48	0.12	0.14	0.42	0.33
MOSSE	0.47	0.0	0.0	0.01	0.44
DECISION TREE	0.86	0.83	0.60	0.78	0.85
RANDOM FOREST	0.87	0.85	0.59	0.79	0.86
LINEAR	0.82	0.77	0.59	0.72	0.82
RIDGE	0.82	0.77	0.59	0.72	0.82
XGBOOST	0.86	0.84	0.60	0.78	0.85
KNN	0.87	0.86	0.57	0.78	0.85
POLYNOMIAL	0.87	0.80	0.60	0.79	0.86
PLS	0.71	0.71	0.26	0.12	0.71

Table 5.8: Expected Average Overlap comparison of different tracking algorithms

TRACKER	Twinnings	Girl	Freeman4	David	Coupon
BOOSTING	0.53	0.29	0.10	0.21	0.90
KCF	0.21	0.02	0.04	0.02	0.46
CSRT	0.69	0.54	0.12	0.14	0.86
MIL	0.47	0.43	0.08	0.07	0.76
TLD	0.33	0.49	0.00	0.02	0.19
MEDIAN FLOW	0.46	0.18	0.10	0.11	0.87
MOSSE	0.0	0.02	0.00	0.12	0.88
DECISION TREE	0.89	0.81	0.74	0.81	0.96
RANDOM FOREST	0.90	0.82	0.74	0.80	0.96
LINEAR	0.85	0.79	0.60	0.55	0.93
RIDGE	0.85	0.79	0.60	0.55	0.93
XGBOOST	0.89	0.81	0.74	0.81	0.96
KNN	0.90	0.83	0.69	0.75	0.95
POLYNOMIAL	0.86	0.81	0.73	0.65	0.94
PLS	0.53	0.70	0.14	0.52	0.78

Table 5.9: Expected Average Overlap comparison of different tracking algorithms

5.2.5 Success Rate :

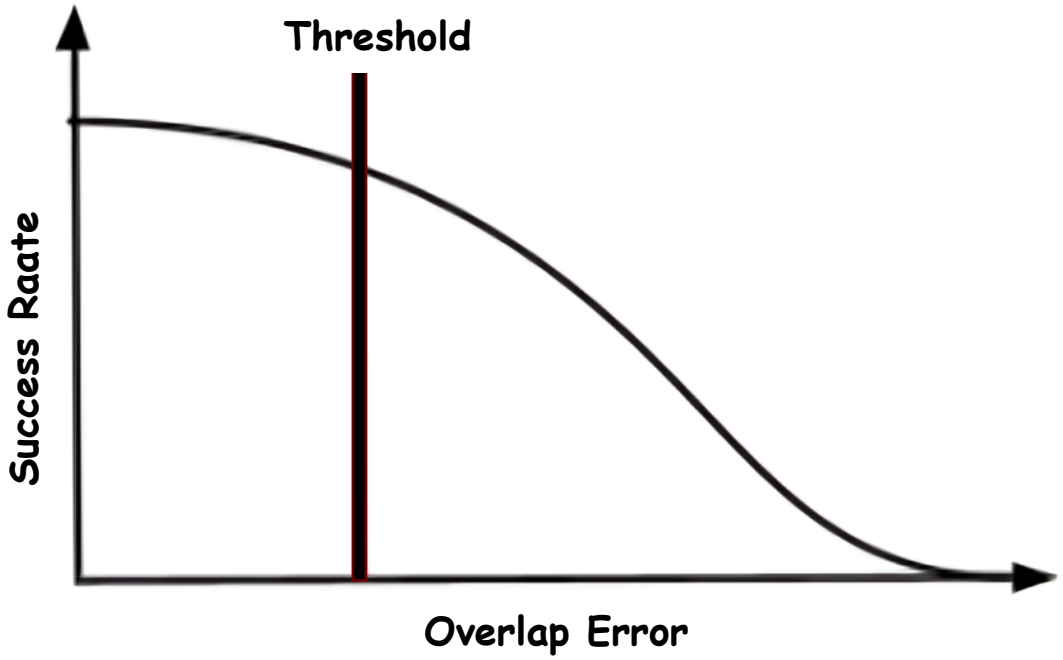


Figure 5.17: An illustration of Success Rate calculation

The success rate of an object tracking algorithm is calculated as the ratio of frames in which the algorithm successfully tracks the object to the total number of frames in a video sequence. The formula for the success rate (SR) is as follows:

$$Success\ Rate = \frac{No.\ of\ Frame\ with\ successfull\ Tracking}{Total\ no.\ of\ Frame} \tag{5.7}$$

This formula provides a percentage that represents the algorithm's effectiveness in tracking the object throughout the entire video sequence. The result shown in Table 5.3 indicates that the success rate of My proposed model performs far better than the deep learning tracker. Four of My proposed meta-learning models such as Decision Tree, Random Forest, KNN and XGBOOST, produce exceptional results in terms of success rate, with a remarkable score of 0.79, 0.74, 0.62 and 0.53 respectively Although the table 5.10 and table 5.11 illustrate the success rate comparison of proposed model in different video sequences of OTB 50 dataset..

TRACKER	Skater	GYM	KiteSurf	Dog	Dancer
BOOSTING	0.72	0.05	0.27	0.18	0.78
KCF	0.05	0.01	0.18	0.02	0.12
CSRT	0.88	0.01	0.32	0.14	0.95
MIL	0.82	0.17	0.40	0.11	0.7
TLD	0.16	0.04	0.39	0.42	0.5
MEDIAN FLOW	0.33	0.03	0.19	0.41	0.27
MOSSE	0.40	0.0	0.0	0.01	0.63
DECISION TREE	0.99	0.99	0.73	0.99	0.99
RANDOM FOREST	0.99	0.99	0.71	0.99	0.99
LINEAR	0.99	0.98	0.77	0.96	0.99
RIDGE	0.99	0.98	0.78	0.96	0.99
XGBOOST	0.99	0.99	0.74	0.99	0.99
KNN	0.99	0.99	0.69	0.97	0.99
POLYNOMIAL	0.99	0.99	0.74	0.99	0.99
PLS	0.94	0.93	0.16	0.07	0.96

Table 5.10: Success Rate comparison of different tracking algorithms at IOU = 0.5

TRACKER	Twinnings	Girl	Freeman4	David	Coupon
BOOSTING	0.55	0.23	0.13	0.08	0.97
KCF	0.19	0.02	0.05	0.03	0.49
CSRT	0.85	0.58	0.15	0.09	0.98
MIL	0.23	0.47	0.06	0.03	0.98
TLD	0.39	0.54	0.00	0.00	0.15
MEDIAN FLOW	0.39	0.16	0.10	0.05	0.96
MOSSE	0.0	0.02	0.00	0.05	0.97
DECISION TREE	0.99	0.99	0.98	0.99	0.99
RANDOM FOREST	0.99	0.99	0.95	0.99	0.99
LINEAR	0.99	0.99	0.74	0.66	0.99
RIDGE	0.99	0.99	0.74	0.66	0.99
XGBOOST	0.99	0.99	0.97	0.99	0.99
KNN	0.99	0.99	0.86	0.91	0.99
POLYNOMIAL	0.99	0.99	0.97	0.82	0.99
PLS	0.52	0.91	0.03	0.56	0.91

Table 5.11: Success Rate comparison of different tracking algorithms at IOU = 0.5

5.3 Ablation Study

This section describes the different aspects of the models and the intuition behind the model selection of the research work. I have done three ablations on the proposed model such as, a study on different combinations of the base tracker, a study on different meta-learners.

5.3.1 A study on different combinations of the base tracker

Table 5.12: RMSE and Frame Rate of the combination of based trackers. C1 represents the combination of MEDIANFLOW and MOOSE. C2 represents the combination of MOOSE and KCF. C3 represents the combination of KCF and MIL. C4 represents the combination of MIL, BOOSTING, and CSRT. C5 represents the combination of CSRT, TLD, and MEDIANFLOW. C6 represents the combination of MIL, BOOSTING, CSRT, and TLD. C7 represents the combination of CSRT, MIL, and TLD.

Model		**LINEAR**	**RIDGE**	**Decision**	**Random**
C1	**RMSE**	140.38	140.04	132.45	122.43
	Frame Rate	430.47	326.84	467.76	72.73
C2	**RMSE**	134.62	134.30	124.72	119.86
	Frame Rate	199.96	183.36	209.17	67.75
C3	**RMSE**	94.42	94.33	118.82	100.70
	Frame Rate	13.85	14.18	12.95	12.28
C4	**RMSE**	105.61	105.32	99.78	83.34
	Frame Rate	8.45	9.06	9.58	8.42
C5	**RMSE**	91.50	91.44	149.59	117.81
	Frame Rate	15.19	16.94	15.56	13.20
C6	**RMSE**	99.24	98.99	103.55	84.95
	Frame Rate	8.94	9.41	8.42	8.84
C7	**RMSE**	80.95	80.97	105.41	84.42
	Frame Rate	15.80	15.47	15.97	13.47

Model		XGBOOST	KNN	Polynomial	PLS
C1	**RMSE**	124.81	163.06	143.49	145.40
	Frame Rate	717.73	196.35	257.00	300.64
C2	**RMSE**	112.70	190.64	127.44	139.10
	Frame Rate	122.49	113.11	181.76	287.77
C3	**RMSE**	99.52	118.76	102.65	97.64
	Frame Rate	10.10	13.33	12.57	14.59
C4	**RMSE**	92.64	101.82	139.16	85.85
	Frame Rate	6.96	7.49	8.99	9.19
C5	**RMSE**	113.96	132.21	136.72	101.73
	Frame Rate	12.07	12.62	17.52	14.77
C6	**RMSE**	89.60	107.76	134.69	84.27
	Frame Rate	7.56	7.80	8.60	9.23
C7	**RMSE**	87.89	98.15	109.10	77.6
	Frame Rate	12.16	12.90	14.90	15.13

Table 5.13: RMSE and Frame Rate of different Proposed Meta-learner.

The Table 5.12 represents the analysis of different combinations of the base models. In this analysis, the C1 represents the combination of MEDIANFLOW and MOOSE, which is a fast processing tracker but very low accuracy. The second column C2 represents the combination of MOOSE and KCF Tracker which improve the accuracy according to the Root Mean Squared value. The combination of C1 is fast compare to the combination of C2. The third column C3 represents the combinations of KCF and MIL, where accuracy is improved due to MIL but still struggling. The next column C4 represents the combination of MIL, BOOSTING, and CSRT, where the top three accurate-based models are used. The next column C5 represents the combination of CSRT, TLD, and MEDIANFLOW. The column C6 represents the combination of MIL, BOOSTING, CSRT, and TLD trackers. Under occlusion, the TLD tracker performs the best result, which affects the accuracy of the proposed model shown in C5 and C6. In that way I move toward slow processing but more accurate and robust tracking results. In My proposed model, I mainly used CSRT, MIL, and TLD shown in column C7, which is more accurate with sufficient processing time. As the C7 combination shows better performance I have used this combination as the final tracker in the research work.

5.3.2 A study on different meta-learners

In this ablation study, I systematically examine the distinctive contributions of My meta-learner in object tracking. By applying the predicted tracker results on different learners such as the LINEAR regression, RIDGE regression, Decision Tree regression model, Random

Forest regression model, XGBOOST regression model, KNN regression model, Polynomial Regression Model and PLS Regression Model. Table 5.13 shows the performance of the different meta-learners on different combinations of base trackers in terms of RMSE and frame rate. In C7 combination PLS meta-learner gives better accuracy in terms of RMSE value and Decision Tree meta-learner provides slightly better Frame rate than other models.

Table 5.14: RMSE Value of proposed Models in the initial and final training process.

Model	**Without Feedback**	**With Feedback**
LINEAR	81.28	80.95
Ridge	81.27	80.97
Decision	103.42	105.41
Random	82.95	84.42
XGBOOST	90.04	87.89
KNN	112.99	98.15
Polynomial	103.24	109.10
PLS	79.96	77.6

5.3.3 A study on the effect of feedback in the learning process

In this study, I tested the proposed model performance on two seniors, such as the model training with the previous frame prediction as feedback and without the previous frame prediction. Table 5.14, In the first column, I present the performance of the model without feedback, while the second column represents the training RMSE value with feedback. I have used the C7 combination as it gives the best result. Table 5.14 shows the model without feedback has very little higher RMSE than the model with feedback. Without feedback, the Decision Tree and Random Forest meta-learner gives best performance but with feedback Linear, Ridge, XGBOOST, KNN and PLS meta-learner provides best performance.

Chapter 6

Conclusion And Future Works

In this study, I aimed to develop a reliable real-time object tracking system that leverages the benefits of machine learning-based trackers over deep learning-based approaches. The proposed method demonstrated significant improvements in success rate, reduced object tracking error, and enhanced model accuracy when compared to existing tracking algorithms. By combining various conventional trackers, I achieved an overall improvement in tracking performance. My model outperformed previous tracking systems in real-time scenarios, with a frame rate of 15.97.

However, there are certain limitations to My proposed model. It is primarily designed for tracking single objects and has a higher frame rate than the current deep learning models, which may impact its performance in real-time applications. In the future, I plan to focus on improving the model's frame rate to ensure seamless performance in real-time scenarios and explore the possibility of extending the system to track multiple objects simultaneously.

LIST OF PUBLICATION

Patents

- A. Ali, S. Sk, **B. Das**, et al., "A real-time sign language translator device," Application No. : 202431007372, Journal No. : 08/2024, Feb. 2024.
- A. Ali, **B. Das**, S. Tripathi, et al., "An embedded system based real-time heart disease detection device," Application No. : 202431015093, Journal No. : 11/2024, Feb. 2024.

Conference

- A. Ali, **B. Das**, S. Sk, S. Ganguli, and S. S. Chaudhuri, "Arabic sign language recognition and voice translation device using embedded machine learning," in *2nd Doctoral Symposium on Human Centered Computing (HUMAN 2024)*.

Journal

- **Bibek Das**, Asfak. Ali, S. Sk, S. Ganguli, and S. S. Chaudhuri, "**Development of Efficient Object Tracking Algorithm using Machine Learning-based Meta-Learners**".

Copyrights

- A. Ali, H. Shaw, S. Ganguly, **B. Das**, and S. Chaudhuri, Hazyvidsegnet: A robust dataset for video object segmentation and tracking in hazy weather conditions, Jan. 2024.
- A. Ali, H. Shaw, S. Sk, **B. Das**, and S. Chaudhuri, Rainsedtrack dataset: Video object segmentation and tracking in rainy weather, Jan. 2024.

Bibliography

[1] Norman M. Abramson, David J. Braverman, and George S. Sebestyen. "Pattern recognition and machine learning". In: *IEEE Trans. Inf. Theory* 9 (1963), pp. 257–261. URL: https://api.semanticscholar.org/CorpusID:42212211.

[2] Ehsan Akbari Sekehravani, Eduard Babulak, and Mehdi Masoodi. "Flying object tracking and classification of military versus nonmilitary aircraft". In: *Bulletin of Electrical Engineering and Informatics* 9 (Aug. 2020), pp. 1394–1403. DOI: 10.11591/eei.v9i4.1843.

[3] S. Avidan. "Ensemble tracking". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. 2005, 494–501 vol. 2. DOI: 10.1109/CVPR.2005.144.

[4] Shai Avidan. "Support vector tracking".. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 1 (2001). 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition ; Conference date: 08-12-2001 Through 14-12-2001, pp. I184–I191. ISSN: 1063-6919.

[5] Qinxun Bai et al. "Randomized Ensemble Tracking." In: *ICCV*. IEEE Computer Society, 2013, pp. 2040–2047. ISBN: 978-1-4799-2839-2. URL: http://dblp.uni-trier.de/db/conf/iccv/iccv2013.html#BaiWSBM13.

[6] Abhishek Balasubramaniam and Sudeep Pasricha. "Object Detection in Autonomous Vehicles: Status and Open Challenges". In: *CoRR* abs/2201.07706 (2022). arXiv: 2201.07706. URL: https://arxiv.org/abs/2201.07706.

[7] Luca Bertinetto et al. "Fully-Convolutional Siamese Networks for Object Tracking". In: *ArXiv* abs/1606.09549 (2016). URL: https://api.semanticscholar.org/CorpusID:14309034.

[8] David S. Bolme, Bruce A. Draper, and J. Ross Beveridge. "Average of Synthetic Exact Filters". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 2105–2112. URL: https://api.semanticscholar.org/CorpusID:6098097.

[9] Yan Chen et al. "3D object tracking via image sets and depth-based occlusion detection". In: *Signal Processing* 112 (2015). Signal Processing and Learning Methods for 3D Semantic Analysis, pp. 146–153. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2014.08.046. URL: https://www.sciencedirect.com/science/article/pii/S0165168414004204.

[10] Zhe Chen, Zhibin Hong, and Dacheng Tao. "An Experimental Survey on Correlation Filter-based Tracking". In: *ArXiv* abs/1509.05520 (2015). URL: https://api.semanticscholar.org/CorpusID:11832048.

[11] Xu Cheng et al. "Object Tracking via Temporal Consistency Dictionary Learning". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.4 (2017), pp. 628–638. DOI: 10.1109/TSMC.2016.2618749.

[12] Martin Danelljan et al. "Convolutional Features for Correlation Filter Based Visual Tracking". In: Dec. 2015, pp. 621–629. DOI: 10.1109/ICCVW.2015.84.

[13] Naa Dardagan et al. "Multiple Object Trackers in OpenCV: A Benchmark". In: *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*. 2021, pp. 1–6. DOI: 10.1109/ISIE45552.2021.9576367.

[14] Dattathreya et al. "Keypoint-based object tracking using modified median flow". In: *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. 2016, pp. 1–2. DOI: 10.1109/ICCE-Asia.2016.7804741.

[15] Holger Dettki, Göran Ericsson, and Lars Edenius. "REAL-TIME MOOSE TRACKING: AN INTERNET BASED MAPPING APPLICATION USING GPS/GSM-COLLARS IN SWEDEN". In: *Alces: A Journal Devoted to the Biology and Management of Moose* 40 (Jan. 2004), pp. 13–21. URL: https://alcesjournal.org/index.php/alces/article/view/427.

[16] Weihao Gan et al. "Object tracking with temporal prediction and spatial refinement (TPSR)". In: *Image Commun.* 47.C (Sept. 2016), pp. 303–312. ISSN: 0923-5965. DOI: 10.1016/j.image.2016.07.005. URL: https://doi.org/10.1016/j.image.2016.07.005.

[17] Helmut Grabner, Michael Grabner, and Horst Bischof. "Real-Time Tracking via On-line Boosting". In: vol. 1. Jan. 2006, pp. 47–56. DOI: 10.5244/C.20.6.

[18] Helmut Grabner, Christian Leistner, and Horst Bischof. "Semi-supervised On-Line Boosting for Robust Tracking". In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 234–247. ISBN: 978-3-540-88682-2.

[19] Bohyung Han et al. "Sequential Kernel Density Approximation and Its Application to Real-Time Visual Tracking". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.7 (2008), pp. 1186–1197. DOI: 10.1109/TPAMI.2007.70771.

[20] João F. Henriques et al. "Exploiting the Circulant Structure of Tracking-by-Detection with Kernels". In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 702–715. ISBN: 978-3-642-33765-9.

[21] João F. Henriques et al. "High-Speed Tracking with Kernelized Correlation Filters". In: *CoRR* abs/1404.7584 (2014). arXiv: 1404.7584. URL: http://arxiv.org/abs/1404.7584.

[22] Seunghoon Hong et al. "Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 597–606. URL: https://proceedings.mlr.press/v37/hong15.html.

[23] Yang Hua, Karteek Alahari, and Cordelia Schmid. "Online Object Tracking with Proposal Selection". In: *CoRR* abs/1509.09114 (2015). arXiv: 1509.09114. URL: http://arxiv.org/abs/1509.09114.

[24] Omar Javed and Mubarak Shah. "Tracking and Object Classification for Automated Surveillance". In: vol. 2353. May 2002, pp. 343–357. ISBN: 978-3-540-43748-2. DOI: 10.1007/3-540-47979-1_23.

[25] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. "P-N learning: Bootstrapping binary classifiers by structural constraints". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 49–56. DOI: 10.1109/CVPR.2010.5540231.

[26] Mahmud Kazi, Abdullah Nahid, and Abdullah Mamun. "Implementation Of Vision Based Object Tracking Robot". In: May 2012. DOI: 10.1109/ICIEV.2012.6317485.

[27] Matej Kristan et al. "The Visual Object Tracking VOT2015 Challenge Results". In: *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. 2015, pp. 564–586. DOI: 10.1109/ICCVW.2015.79.

[28] Siyi Li and D. Y. Yeung. "Visual Object Tracking for Unmanned Aerial Vehicles: A Benchmark and New Motion Models". In: *AAAI Conference on Artificial Intelligence*. 2017. URL: https://api.semanticscholar.org/CorpusID:12949500.

[29] Xiuzhi Li et al. "Visual Tracking Based on Convolutional Neural Network". In: *2018 Chinese Automation Congress (CAC)*. 2018, pp. 4061–4066. DOI: 10.1109/CAC.2018.8623581.

[30] Yanli Liu et al. "Long Short-Term Memory Networks Based on Particle Filter for Object Tracking". In: *IEEE Access* 8 (2020), pp. 216245–216258. DOI: 10.1109/ACCESS.2020.3041294.

[31] Alan Lukezic et al. "Discriminative Correlation Filter With Channel and Spatial Reliability". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[32] Alan Lukezic et al. "Discriminative Correlation Filter with Channel and Spatial Reliability". In: *CoRR* abs/1611.08461 (2016). arXiv: 1611.08461. URL: http://arxiv.org/abs/1611.08461.

[33] Alan Lukei, Luka ehovin Zajc, and Matej Kristan. "Deformable Parts Correlation Filters for Robust Visual Tracking". In: *IEEE Transactions on Cybernetics* PP (May 2016). DOI: 10.1109/TCYB.2017.2716101.

[34] Nikos Paragios and Rachid Deriche. "Geodesic Active Regions and Level Set Methods for Supervised Texture Segmentation". In: *International Journal of Computer Vision* 46 (2002), pp. 223–247. URL: https://api.semanticscholar.org/CorpusID:18836072.

[35] Horst Possegger, Thomas Mauthner, and Horst Bischof. "In defense of color-based model-free tracking". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 2113–2120. URL: https://api.semanticscholar.org/CorpusID:9470021.

[36] Weijian Ruan et al. "Multi-Correlation Filters With Triangle-Structure Constraints for Object Tracking". In: *Trans. Multi.* 21.5 (May 2019), pp. 1122–1134. ISSN: 1520-9210. DOI: 10.1109/TMM.2018.2872897. URL: https://doi.org/10.1109/TMM.2018.2872897.

[37] Pedro Senna, Isabela Neves Drummond, and Guilherme Sousa Bastos. "Real-Time Ensemble-Based Tracker with Kalman Filter". In: *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. 2017, pp. 338–344. DOI: 10.1109/SIBGRAPI.2017.51.

[38] Zahra Soleimanitaleb, Mohammad Ali Keyvanrad, and Ali Jafari. "Improved MDNET Tracker in Better Localization Accuracy". In: *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)*. 2020, pp. 221–226. DOI: 10.1109/ICCKE50421.2020.9303727.

[39] B. Stenger et al. "Topology free hidden Markov models: application to background modeling". In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 1. 2001, 294–301 vol.1. DOI: 10.1109/ICCV.2001.937532.

[40] Jack Valmadre et al. "End-to-End Representation Learning for Correlation Filter Based Tracking". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5000–5008. URL: https://api.semanticscholar.org/CorpusID:10520310.

[41] Tomás Vojír, Jana Noskova, and Jiri Matas. "Robust scale-adaptive mean-shift for tracking". In: *Pattern Recognition Letters*. 2013. URL: https://api.semanticscholar.org/CorpusID:6047702.

[42] Eric A. Wan and Rudolph van der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)* (2000), pp. 153–158. URL: https://api.semanticscholar.org/CorpusID:13992571.

[43] N. Wang and Dit-Yan Yeung. "Ensemble-based tracking: Aggregating crowdsourced structured time series data". In: *31st International Conference on Machine Learning, ICML 2014* 4 (Jan. 2014), pp. 2807–2817.

[44] Naiyan Wang, Jingdong Wang, and Dit-Yan Yeung. "Online Robust Non-negative Dictionary Learning for Visual Tracking". In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 657–664. DOI: 10.1109/ICCV.2013.87.

[45] Naiyan Wang and Dit-Yan Yeung. "Ensemble-based tracking: aggregating crowdsourced structured time series data". In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China: JMLR.org, 2014, pp. II1107II–1115.

[46] Xiaoyu Wang, Tony X. Han, and Shuicheng Yan. "An HOG-LBP human detector with partial occlusion handling". In: IEEE, 2009, pp. 32–39. DOI: 10.1109/ICCV.2009.5459207.

[47] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. "Object Tracking Benchmark". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1834–1848. DOI: 10.1109/TPAMI.2014.2388226.

[48] Yanchun Xie et al. "Correlation Filter Selection for Visual Tracking Using Reinforcement Learning". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.1 (2020), pp. 192–204. DOI: 10.1109/TCSVT.2018.2889488.

[49] Alper Yilmaz, Omar Javed, and Mubarak Shah. "Object tracking: A survey". In: *ACM Comput. Surv.* 38.4 (Dec. 2006), 13–es. ISSN: 0360-0300. DOI: 10.1145/1177352.1177355. URL: https://doi.org/10.1145/1177352.1177355.

[50] Fan Zhang. "Moving Object Tracking Method Based on SVM and Meanshift Tracking Algorithm". In: *Proceedings of the 2022 6th International Conference on Deep Learning Technologies*. ICDLT '22. Xi'an, China: Association for Computing Machinery, 2022, pp. 84–88. ISBN: 9781450396936. DOI: 10.1145/3556677.3556701. URL: https://doi.org/10.1145/3556677.3556701.

[51] Jin Zhang et al. "Robust Human Tracking Algorithm Applied for Occlusion Handling". In: *2010 Fifth International Conference on Frontier of Computer Science and Technology* (2010), pp. 546–551. URL: https://api.semanticscholar.org/CorpusID:12204339.

[52] Zhen et al. "A Visual Object Tracking Algorithm Based on Improved TLD". In: *Algorithms* 13 (Jan. 2020), p. 15. DOI: 10.3390/a13010015.

[53] Gao Zhu, Fatih Porikli, and Hongdong li. "Tracking Randomly Moving Objects on Edge Box Proposals". In: (July 2015).