# Enhancing Underwater Fish Tracking through Ensemble Methods and Autonomous Reinitialization

A Thesis submitted to the faculty of  
San Francisco State University  
In partial fulfillment of  
the requirements for  
the Degree

Master of Science  
In  
Computer Science

by  
  
Ekarat Buddharuksa  
  
San Francisco, California  
  
December 2025

Copyright by  
Ekarat Buddharuksa  
2025

## Certification of Approval

I certify that I have read Enhancing Underwater Fish Tracking through Ensemble Methods and Autonomous Reinitialization by Ekarat Buddharuksa, and that in my opinion this work meets the criteria for approving a thesis submitted in partial fulfillment of the requirement for the degree Master of Science in Computer Science at San Francisco State University.

Kazunori Okada,Ph.D.

Professor,

Thesis Committee Chair

E. Wes Bethel,Ph.D.

Associate Professor

Sara El Alaoui, Ph.D.

Assistant Professor

## Abstract

The thesis presents a semi-autonomous ensemble-based tracking framework for underwater fish tracking. By combining multiple traditional OpenCV trackers with Kalman filters and fusing their predictions using Covariance Intersection, the system improves stability and robustness in visually challenging environments. It employs Mahalanobis distance for outlier filtering. The framework is evaluated on annotated sequences from the DeepFish dataset with a simulated automation with the ground-truth. The results demonstrate enhanced accuracy, zero failure cases, and no need for manual reinitialization at the fused output level.

## Acknowledgements

I would like to express my deepest gratitude to those who have supported and guided me throughout the completion of this thesis.

First, I want to sincerely thank Professor Kazunori Okada for his steady guidance, insightful feedback, and constant encouragement since 2023. His mentorship has been instrumental in helping me grow and overcome the challenges of this research.

I also truly appreciate Professor Wes Bethel for his support and being part of the committee during this important stage. Special thanks to Professor Sara El Alaoui for stepping in as a committee member, her flexibility and support made it possible for me to complete this thesis on time.

I’m especially thankful to Chris Huber, Yoshimasa Iwano, Aaron Kuo, and Marie Shimizu, whose dedication to their own research inspired me in many ways. Our conversations and shared interests helped broaden my perspective and deepened my understanding of computational methods across different domains.

Finally, to my friends and family, thank you for your endless patience, encouragement, and belief in me. Your support during the most difficult moments reminded me to keep moving forward.

This thesis is not just the result of my own efforts, but a reflection of the kind and collaborative environment I’ve been lucky to be part of.

Table of Contents

[Enhancing Underwater Fish Tracking through Ensemble Methods and Autonomous Reinitialization i](#_Toc197875815)

[Certification of Approval 2](#_Toc197875816)

[Abstract 3](#_Toc197875817)

[Acknowledgements v](#_Toc197875818)

[List of Tables viii](#_Toc197875819)

[List of Figures ix](#_Toc197875820)

[Chapter 1: Introduction 1](#_Toc197875821)

[1.1 Problem: The Challenge of Object Tracking in Underwater Environments 1](#_Toc197875822)

[1.2 Computer Vision for Object Tracking 1](#_Toc197875823)

[1.3 Existing Solutions in Object Tracking 2](#_Toc197875824)

[1.4 Technical proposal 4](#_Toc197875825)

[1.5 Dataset and Application 6](#_Toc197875826)

[1.6 Thesis structure 7](#_Toc197875827)

[Chapter 2: Literature Review 8](#_Toc197875828)

[2.1 Evolution of Object Tracking Techniques 8](#_Toc197875829)

[2.2 Object Detection Framework 14](#_Toc197875830)

[2.3 Public Data for Underwater Fish Analysis 15](#_Toc197875831)

[2.4 Multi-Sensor Fusion in Object Tracking 16](#_Toc197875832)

[2.5 Detection and Tracking of Objects in Underwater Video 18](#_Toc197875833)

[2.6 Conclusion 19](#_Toc197875834)

[Chapter 3: Methodology 21](#_Toc197875835)

[3.1 Introduction 21](#_Toc197875836)

[3.2 Proposed Method 22](#_Toc197875837)

[3.3 Evaluation Metric 29](#_Toc197875838)

[3.4 Data used 32](#_Toc197875839)

[3.5 Computational environment 36](#_Toc197875840)

[3.6 Conclusion 36](#_Toc197875841)

[Chapter 4: Experimental results 38](#_Toc197875842)

[4.1 Introduction 38](#_Toc197875843)

[4.2 Validation design 38](#_Toc197875844)

[4.3 Result 40](#_Toc197875845)

[4.4 Conclusion 46](#_Toc197875846)

[Chapter 5: Discussion 48](#_Toc197875847)

[5.1 Introduction 48](#_Toc197875848)

[5.2 Performance of the Fused Tracker 48](#_Toc197875849)

[5.3 Role of Kalman Filtering 51](#_Toc197875850)

[5.4 Role of Outliner filtering utilizing Mahalanobis Distance 51](#_Toc197875851)

[5.5 Role of Covariance Intersection (CI) 54](#_Toc197875852)

[5.6 Purpose of Simulated Automation with Ground-Truth policy 55](#_Toc197875854)

[5.7 Validation of Simulated Automation with Ground-Truth Policy and Its Influence on Fused Output 55](#_Toc197875855)

[5.8 Implications for Real-World Deployment 59](#_Toc197875856)

[5.9 Conclusion 60](#_Toc197875857)

[Chapter 6: Conclusion and Future Work 61](#_Toc197875858)

[6.1 Conclusion 61](#_Toc197875859)

[6.2 Future Work 62](#_Toc197875860)

[Bibliography 65](#_Toc197875861)

[Appendix 71](#_Toc197875862)

## List of Tables

[Table 4‑1: 9866\_acanthopagrus\_palmaris Tracker Evaluation 40](#_Toc197875863)

[Table 4‑2: 9862\_Acanthopagrus\_palmaris Tracker Evaluation 41](#_Toc197875864)

[Table 4‑3: 7482\_F3 Tracker Evaluation 41](#_Toc197875865)

[Table 4‑4: 9866\_acanthopagrus\_palmaris Kalman Evaluation 42](#_Toc197875866)

[Table 4‑5: 9862\_Acanthopagrus\_palmaris Kalman Evaluation 43](#_Toc197875867)

[Table 4‑6: 7482\_F3 Kalman Evaluation 43](#_Toc197875868)

[Table 5‑1: Number of filtered trackers and performance of the fused tracker on Video\_1 with 1.5-Sigma rule in Kalman setting 53](#_Toc197875869)

[Table 5‑2: Number of filtered out trackers and performance of the fused tracker on Video\_1 with 2-Sigma rule in Kalman setting 53](#_Toc197875870)

[Table 5‑3: Video\_1 Result with 10-frames rule 58](#_Toc197875871)

[Table 5‑4: Video\_1 Result with 5-frames rule 58](#_Toc197875872)

[Table 5‑5: Video\_1 Result with 1-frames rule 59](#_Toc197875873)

## List of Figures

[Figure 1‑1: System Overview 4](#_Toc197876314)

[Figure 3‑1: Failure-based reinitialization logic overview 27](#_Toc197876315)

[Figure 3‑2: 9866\_acanthop agrus\_palmaris (Video\_1) from DeepFish dataset#1 34](#_Toc197876316)

[Figure 3‑3: 9862\_Acanthopagrus\_palmaris (Video\_2) from DeepFish dataset#2 34](#_Toc197876317)

[Figure 3‑4: 7482\_F3 (Video\_3) from DeepFish dataset#3 34](#_Toc197876318)

[Figure 3‑5: Roboflow annotation tools 35](#_Toc197876319)

[Figure 5‑1: Output Frame from Video\_1 Demonstrating Fused Tracker Stability Despite Individual Tracker Failures 49](#_Toc197876320)

[Figure A‑1: Video\_1 RMSE over time for Tracker 71](#_Toc197876321)

[Figure A-2: Video\_1 RMSE over time for Kalman 71](#_Toc197876322)

[Figure A‑3: Video\_1 Euclidean distance over time for Kalman 72](#_Toc197876323)

[Figure A-4: Video\_1 Euclidean distance over time for Tracker 72](#_Toc197876324)

[Figure A-5: Video\_1 IoU over time for Kalman 73](#_Toc197876325)

[Figure A‑6: Video\_1 IoU over time for Tracker 73](#_Toc197876326)

[Figure A‑7: Video\_2 RMSE over time for Kalman 74](#_Toc197876327)

[Figure A‑8: Video\_2 RMSE over time for Tracker 74](#_Toc197876328)

[Figure A‑9: Video\_2 Euclidean distance over time for Kalman 75](#_Toc197876329)

[Figure A-10: Video\_2 Euclidean distance over time for Tracker 75](#_Toc197876330)

[Figure A‑11: Video\_2 IoU over time for Tracker 76](#_Toc197876331)

[Figure A‑12: Video\_2 IoU over time for Kalman 76](#_Toc197876332)

[Figure A‑13: Video\_3 RMSE over time for Kalman 77](#_Toc197876333)

[Figure A‑14: Video\_3 RMSE over time for Tracker 77](#_Toc197876334)

[Figure A‑15: Video\_3 Euclidean distance over time for Kalman 78](#_Toc197876335)

[Figure A‑16: Video\_3 Euclidean distance over time for Tracker 78](#_Toc197876336)

[Figure A‑17: Video\_3 IoU over time for Tracker 79](#_Toc197876337)

[Figure A‑18: Video\_3 IoU over time for Kalman 79](#_Toc197876338)

# Chapter 1: Introduction

### Problem: The Challenge of Object Tracking in Underwater Environments

Object tracking in dynamic environments is a well-established problem in computer vision, playing a crucial role in applications such as surveillance, autonomous navigation, and wildlife monitoring. However, tracking objects underwater presents unique challenges that distinguish it from tracking in terrestrial environments. Poor visibility caused by light refraction, turbidity, and suspended particles reduces the clarity of underwater images. In addition, dynamic water currents and environmental disturbances cause frequent occlusions, further complicating the tracking process. Tracking fish in underwater environments is particularly difficult due to their erratic movements, which result in rapid and unpredictable trajectory changes. The presence of marine vegetation and complex background textures further increases the likelihood of confusing what object to track. Traditional object tracking algorithms, which perform well in controlled conditions, often fail when applied to underwater settings. These failures manifest as tracking drift, loss of tracking, and unreliable motion estimation. As a result, there is a need for tracking systems that can address these unique challenges and provide robust, adaptive solutions for underwater single fish tracking.

### 1.2 Computer Vision for Object Tracking

Computer vision techniques have been widely used for object detection and tracking in various domains. Over the years, tracking methodologies have evolved to address challenges such as motion blur, occlusions, and scale variations. Optical flow methods, such as Lucas-Kanade [1], and predictive filtering techniques like the Kalman filter [2] have been foundational in tracking research, particularly under linear motion assumptions. More recently, machine learning-based tracking methods have significantly improved robustness by adapting to complex object appearances and motion patterns. However, the introduction of deep learning-based tracking has further enhanced tracking accuracy at the cost of computational efficiency [3].

### 1.3 Existing Solutions in Object Tracking

Tracking algorithms in computer vision can be broadly categorized into early/traditional methods, machine learning- based method, and deep learning-based methods. Each of these approaches has contributed to advancements in object tracking, but they also present limitations, particularly when applied to underwater fish tracking, where environmental conditions are unpredictable, and object appearances change dynamically.

Tradition tracking methods relied on fundamental motion estimation techniques such as optical flow, Correlation filter, and Kalman filter. Optical flow methods, including Lucas-Kanade [1], estimate the displacement of pixel intensities between consecutive frames to track object motion. The Kalman filter [2], a predictive modeling approach, improved tracking by estimating object positions while accounting for motion noise. However, it assumes linear motion, making it less effective in complex environments where objects move unpredictably. Correlation filter-based trackers (CSRT [4], MOSSE [5], MedianFlow [6]) rely on template matching and feature tracking for object localization. Although these methods improve tracking in structured environments, they remain vulnerable to background clutter, object drift, and sudden occlusions.

Machine learning-based trackers, such as Boosting [7], MIL [8], and KCF [9], improve adaptability by learning object features over time. Boosting and MIL use adaptive classification, while KCF enhances tracking accuracy with correlation filtering. These methods handle occlusions better than traditional trackers but remain prone to tracking drift and degradation over time.

Deep learning-based trackers, including SiamRPN and FairMOT [10, 11], use convolutional neural networks (CNNs) for feature extraction, achieving higher accuracy and robustness. However, they require significant computational power and large datasets, making real-time underwater deployment challenging.

Despite these advances, no single tracker is universally reliable across all scenarios, especially when applied to underwater fish tracking. The unpredictable nature of fish movement and underwater conditions often leads to failure in single tracker systems, such as drift or occlusion. In response to these issues, multi-tracker systems have been explored, where several trackers work together to improve accuracy and resilience. Further details on the specific trackers discussed in this section can be found in Chapter 3.2: Previous Methods.

### 1.4 Technical proposal

A diagram of a system overview
To address the specific challenges of underwater single fish tracking, this thesis proposes a semi-automate multi-tracker system supported by Kalman filters and Covariance Intersection (CI) for fusion. This proposed method aggregates outputs from multiple OpenCV trackers [12], smooths object motion and acquire each tracker covariance using Kalman filters [13, 14, 2], detect outliner using mahalanobis distance [15], and fuses them into a reliable tracking estimate using Covariance Intersection [16, 17].

Figure 1‑1: System Overview

In each video frame, multiple independent OpenCV trackers are used to estimate the fish’s location. To improve temporal consistency and quantify uncertainty, each tracker is paired with a dedicated Kalman filter [2] —explained in Section 3.2.2. The Kalman filter receives the tracker's estimate, smooths the trajectory, and most importantly, generates a covariance matrix that describes the uncertainty of that tracker's current prediction. To further improve the framework, the system applies Mahalanobis distance filtering [15]—mention in 3.2.3— to identify and reject outlier trackers—those whose predictions deviate significantly from the ensemble. This ensures that only statistically consistent trackers contribute to the final fused result.

The remaining estimates are then fused using Covariance Intersection (CI) [17]—mention in 3.2.4. CI performs a nonlinear weighted combination of the valid tracker outputs. The weights are derived from the inverse of the uncertainty—trackers with lower covariance (i.e., more confidence) are given more influence. CI does not require knowledge of the correlation between trackers, making it ideal for fusing estimates from different sources operating on the same input. This fusion process produces a single, robust, and uncertainty-aware estimate of the fish’s location in each frame.

After generating the fused output using Covariance Intersection, we handle tracker failures through a failure-based reinitialization strategy—mention in 3.2.5.1. Each frame, we call .update() on all trackers. If a tracker fails to return a bounding box, it’s flagged and added to a failed list. Once the valid trackers produce a fused estimate, we use this fused location to reinitialize all failed trackers — placing them at the fused position and assigning them the average bounding box size of the successful ones. This automatic recovery ensures that trackers rejoin the ensemble without manual intervention, improving long-term stability and maintaining ensemble diversity.

Finally, in real deployments, trackers are often supported by object detectors for recovery once the tracking is drifted. Since this thesis focuses purely on tracking performance, I simulate that recovery using Simulated Automation with Ground-Truth policy—mention in 3.2.5.2. If a tracker drifts outside the ground truth, I reset it and log the event in a CSV file to ensure repeatability. This mechanism is only for evaluation.

This approach leverages multi-tracker fusion, predictive modeling, and statistical filtering to provide a more accurate, robust, and adaptable solution for underwater fish tracking, making it well-suited for real-world applications in marine biology and ecological research.

### 1.5 Dataset and Application

The primary application of this research is in marine biology and ecological monitoring, where tracking individual fish in their natural habitats can yield valuable insights into behavior, health, and environmental interactions. By automating the tracking process, the system aims to facilitate long-term monitoring while reducing the need for manual annotation, making it a valuable tool for researchers in marine biology and fishery management.

To validate effectiveness of the proposed method toward these goals, a subset of DeepFish dataset [18] was utilized. This dataset contains a diverse collection of underwater images and videos featuring various species and habitats. Selected sequences were manually annotated using RoboFlow, an online annotation platform. Further details on the dataset selection and annotation process are provided in Section 3.4. This setup enables comprehensive testing of the system under diverse and realistic underwater conditions.

### 1.6 Thesis structure

This thesis is structured into six chapters. Chapter 2 provides a literature review, summarizing key research in object tracking, multi-tracker fusion, and underwater computer vision.

Chapter 3 describes the methodology used in this work, including our proposed method discussing multi-tracker system detailing the integration of each tracker, Kalman filtering, Mahalanobis distance-based outlier detection, and Covariance Intersection fusion to improve tracking robustness. It also describes the evaluation metric, and computational environment used to test the proposed method on the subset of DeepFish dataset [18].

Chapter 4 detailing the validation design and presents experimental results, providing the summary of the performance of the fusion framework. Chapter 5 further provides a discussion of the results, role of each component in the framework, and implications for real-world deployment. Finally, Chapter 6 concludes the thesis by summarizing key findings and contributions while outlining future research directions.

# Chapter 2: Literature Review

This section discusses key papers and methodologies that have shaped the field of underwater object tracking, emphasizing the variety of approaches and innovations introduced over the years. This thesis will specifically focus on enhancing OpenCV-Based Trackers by fusing six selected trackers using Kalman filtering, covariance intersection, and outlier detection with Mahalanobis distance to improve the robustness of tracking in challenging environments.

### 2.1 Evolution of Object Tracking Techniques

#### 2.1.1 Traditional Tracking Method

Over the past decades, a wide array of tracking algorithms has been developed, ranging from early model-based techniques to modern data-driven approaches. These methods differ in how they represent object appearance, handle motion uncertainty, and adapt to complex visual conditions.

Traditional tracking algorithms generally rely on handcrafted features and analytical motion models. Among the most prominent examples is the Kalman filter, which operates under the assumption of linear dynamics and Gaussian noise. It provides efficient recursive estimation of the object’s state and is well-suited for tracking objects with predictable motion patterns [2]. Another widely used technique is optical flow, which estimates the apparent displacement of image intensities across consecutive frames. While effective for capturing fine motion details, optical flow is sensitive to visual noise, occlusion, and abrupt object motion [1].

Other classical approaches include Mean-Shift [19]— introduced by Wen and Cai— and CamShift algorithms. These methods track objects by modeling their appearance using color histograms and searching for the most similar distribution in subsequent frames [20]. Although computationally efficient and straightforward to implement, they often perform poorly under significant appearance variation or background clutter. Template matching has also been used historically for object tracking by comparing fixed image patches across frames. However, this method is both computationally intensive and highly susceptible to failure in the presence of occlusion, deformation, or lighting variation.

While these traditional techniques are effective in controlled environments, they generally lack the adaptability and robustness required in dynamic and unconstrained real-world scenarios, such as those found in underwater tracking.

#### 2.1.2 Machine Learning-Based Tracking Method

Machine learning-based tracking methods have significantly advanced the state of the art by leveraging large datasets and learned representations. These approaches eliminate the need for manually crafted features and enable more adaptive tracking. One notable class of models includes Siamese network-based trackers such as SiamFC and SiamRPN introduced by Huang et al. [10]. These models learn a similar function between a reference image and a search region, enabling the tracking task to be formulated as a matching problem. They are recognized for their real-time performance and resilience to changes in object appearance.

Another dominant strategy is tracking-by-detection, where objects are detected in every frame and then linked across time using data association techniques. This approach benefits greatly from recent improvements in object detectors such as YOLO, SSD, and Faster R-CNN [21, 22]. Methods such as Deep SORT [11] extend this strategy by incorporating deep appearance embeddings and motion prediction modules, often powered by Kalman filters, to enhance the reliability of temporal association, especially in crowded or cluttered environments

#### 2.1.3 OpenCV-Based Trackers

This section discusses the key paper explaining the OpenCV object tracking algorithm. The OpenCV legacy tracking API [12] offers object-oriented implementations of popular tracking algorithms, including BOOSTING, MIL, KCF, CSRT, MedianFlow, MOSSE, and TLD. These trackers are integral to the methodology of this thesis. The following list presents some details of these legacy tracking algorithms.

##### Boosting Tracker

The Boosting Tracker [7] is based on the online AdaBoost algorithm, formulates object tracking as a binary classification task. It incrementally trains a strong classifier from a set of weak learners to distinguish the object from the background in real-time. The tracker uses simple features, such as Haar-like patterns, and updates the classifier online as new frames arrive, allowing it to adapt to appearance changes.

Its main strengths lie in adaptability and discriminative feature selection. However, it is prone to drift if misclassifications accumulate during updates. Additionally, the tracker’s performance is sensitive to initial sample selection and may degrade in complex scenes with frequent occlusions or rapid motion.

##### MIL (Multiple Instance Learning) Tracker

The MIL tracker improves upon the Boosting approach by introducing a probabilistic framework for learning from ambiguous or noisy positive samples. Instead of relying on a single sample to represent the object, MIL treats a "bag" of potential samples as positive and uses a discriminative model to infer which instances best represent the object [8]. This strategy increases robustness to labeling errors and minor drifts during updates. However, the MIL tracker may still experience cumulative drift over long sequences, especially in the presence of partial occlusions or significant deformations.

##### KCF (Kernelized Correlation Filters) Tracker

The KCF Tracker [9], utilizes kernelized correlation filters and circulant matrix properties to achieve high-speed tracking with improved accuracy. By operating in the Fourier domain, it significantly reduces computational complexity while enabling the use of non-linear kernels. KCF performs well under moderate appearance changes and is highly efficient for real-time applications. However, it does not handle scale variation natively and may fail under occlusion or rapid object deformation.

##### Median Flow Tracker

The Median Flow Tracker [6] employs a point-tracking approach using the Lucas-Kanade method to estimate optical flow. It evaluates the forward-backward error by tracking points forward in time and then backward to their original positions, identifying inconsistencies that indicate tracking failures. By computing the median of these flow vectors, the tracker achieves robustness to outliers and provides reliable tracking under conditions of smooth and predictable motion. However, it is sensitive to rapid movements, significant occlusions, and abrupt illumination changes, which can lead to tracking failures.

##### MOSSE Tracker

The MOSSE (Minimum Output Sum of Squared Error) Tracker [5] employs adaptive correlation filters to achieve robust and high-speed object tracking. By minimizing the sum of squared errors between the filter output and a desired response, MOSSE creates filters that are adept at distinguishing the target from the background. Operating efficiently in the frequency domain using Fast Fourier Transform (FFT), the tracker can process hundreds of frames per second, making it suitable for real-time applications. MOSSE demonstrates resilience to variations in lighting, scale, and pose, and can detect occlusions by analyzing the peak-to-sidelobe ratio, allowing it to pause and resume tracking as needed. However, its reliance on grayscale features may limit its effectiveness in scenarios requiring discrimination based on color or texture.

##### CSRT Tracker

The CSRT (Channel and Spatial Reliability Tracking) Tracker [4] is an object tracking algorithm implemented in OpenCV, based on the Discriminative Correlation Filter with Channel and Spatial Reliability (CSR-DCF) method. This approach enhances traditional correlation filters by integrating channel reliability and spatial reliability mechanisms, allowing the tracker to focus on the most informative regions of the target object and adapt to changes in appearance. CSRT utilizes features such as Histogram of Oriented Gradients (HOG) [23] and Color Names to build a robust representation of the target. While CSRT offers improved accuracy and robustness against occlusions and non-rigid deformations, it operates at a lower speed compared to some other trackers, making it less suitable for real-time applications requiring high frame rates.

To implement these algorithms in modern computer language, LearnOpenCV [11] provide detailed tutorials and real-world use cases for implementing these algorithms in Python and C++. They also introduce advanced tracking frameworks such as DeepSort and FairMOT, which extend traditional methods using deep learning-based re-identification models.

There are several studies that experimentally compare legacy tracking algorithms. Manzoor et al. [3] conducted a comparative analysis of machine learning-based and deep learning-based object tracking, benchmarking the performance of various algorithms under the same conditions. Their study highlights the advantages and limitations of both approaches, providing valuable insights into the trade-offs between accuracy, computational efficiency, and robustness.

An online article from Broutonlab [23] provides an in-depth explanation of different object tracking algorithms, detailing their theoretical foundations and practical implementations. This resource serves as a basis for the code developed in this thesis, offering a structured reference for integrating OpenCV-based tracking techniques with Kalman filtering and other fusion methods.

Dardagan, et al. [24] evaluated multiple OpenCV tracking algorithms, assessing their strengths and weaknesses across different scenarios. Their benchmark study provides empirical evidence supporting the selection of robust and efficient tracking algorithms for real-world applications, particularly in underwater environments.

Levin, et al. [25] provides an experimental evaluation of OpenCV tracking algorithms applied to ultrasound videos, highlighting their practical limitations and behavior in noisy, real-world imaging conditions.

### 2.2 Object Detection Framework

In practice, the effectiveness of these OpenCV-based tracking algorithms often depends on robust initialization and reinitialization strategies, particularly in challenging conditions such as occlusion, drift, or complete tracking failure. To address these limitations, object detection frameworks can be used to complement tracking by providing reliable object localization when tracking confidence drops. Notably, real-time detectors such as YOLO (You Only Look Once) [21] and SSD (Single Shot Detector) [22] offer fast, one-shot detection capabilities that make them well-suited for integration with OpenCV trackers, enabling automatic reinitialization when tracking is lost. These detection-based aids enhance the robustness of tracking systems, especially in dynamic or cluttered environments. Furthermore, comprehensive surveys by Yilmaz et al. [26] and Wu et al. [27] provide historical context and standardized benchmarks that guide the evaluation and development of both detection and tracking algorithms.

### 2.3 Public Data for Underwater Fish Analysis

The DeepFish dataset, introduced by Saleh et al. [18], provides a diverse collection of underwater fish images and videos for evaluating tracking and detection algorithms. It includes various underwater environments, fish species, and challenging visual conditions such as murky water, occlusions, and low contrast. This dataset has been widely used in benchmarking modern tracking algorithms [28], particularly those employing deep learning and machine learning techniques for robustness in real-world underwater scenarios. In this thesis, the DeepFish dataset serves as the primary evaluation benchmark. The dataset’s diverse environmental conditions make it suitable for testing multi-tracker fusion methods, including Kalman filtering [2], covariance intersection [17], and Mahalanobis distance-based outlier rejection [15].

In parallel, Kezebou et al. [32] introduced the first comprehensive benchmark and dataset specifically designed for underwater object tracking, named UOT32. Their dataset contains 32 underwater videos totaling 24,241 annotated frames, addressing the lack of standardized underwater benchmarks. They demonstrated that state-of-the-art trackers, such as CCOT and ECO, significantly degrade in performance when applied to underwater scenarios, emphasizing the unique challenges posed by distortions like refraction, absorption, and particle interference inherent to underwater visual data. This work also reviewed the UOT100 dataset, introduced by Panetta et al. [29] , which consists of 104 underwater video sequences and more than 74,000 annotated frames derived from both natural and artificial underwater sources. Although UOT32 and UOT100 was not used directly in the experiments, it provides valuable insights into the challenges of underwater tracking and serves as an important reference for understanding performance expectations and evaluation standards in this domain.

### 2.4 Multi-Sensor Fusion in Object Tracking

Multi-sensor fusion plays a critical role in object tracking by combining multiple sources of information to improve accuracy and robustness. Various methods have been explored in literature, particularly in scenarios where individual trackers may provide uncertain or inconsistent results. This section reviews key works related to multi-tracker fusion techniques, including Kalman filtering, Mahalanobis distance-based outlier detection, and covariance intersection.

#### 2.4.1 Kalman Filter-Based Fusion

The Kalman filter [2] is a widely used technique in object tracking and sensor fusion. It models an object’s motion using a state-space representation and continuously updates predictions with new measurements. A practical, hands-on explanation is provided in an online training tutorial by Pierian Training [13], which demonstrates how to implement the Kalman filter in OpenCV for improved tracking performance. This resource illustrates how the filter reduces noise measurement and handles missing observations, contributing to smoother and more reliable object tracking.

The effectiveness of Kalman filtering for multi-tracker fusion has been studied in Anitha et al. [30], where the filter was used to integrate multiple sensor measurements for improved tracking. Their work shows that Kalman filtering enhances tracking stability, particularly in dynamic environments.

#### 2.4.2 Covariance Intersection for Sensor Fusion

Covariance Intersection (CI) is an advanced data fusion technique that merges uncertain measurements without requiring knowledge of their correlations. Forsling et al. [17] provide a comprehensive overview of CI’s evolution over the past 25 years, emphasizing its continued relevance in systems where cross-correlations are unknown or difficult to estimate. Stone Soup [16] explains the theoretical foundations of CI and its application in multi-sensor fusion, showing that it is particularly useful when individual sensor uncertainties vary over time.

Jiang and Xiao [31] proposed a multi-sensor CI Kalman filter for target tracking that eliminates the need for cross-covariance computation, reducing complexity in distributed fusion scenarios. Their simulation demonstrated that the CI fusion filter outperforms local Kalman filters and approaches the accuracy of optimal matrix- and scalar-weighted fusion methods. Moreover, the CI filter recorded the shortest computation time validating its efficiency and practical feasibility in real-time applications.

Spampinato et al. [33] proposed a fish tracking algorithm based on covariance matrices of spatial and appearance features, achieving over 90% accuracy on real-life underwater video data by effectively handling occlusions and environmental variability, outperforming traditional approaches like CAMSHIFT.

#### 2.4.3 Mahalanobis Distance for Outlier Detection

Ghorbani[15] explores the application of Mahalanobis distance in detecting multivariate outliers, emphasizing its effectiveness in identifying deviations in high-dimensional data. This method is widely used in tracking systems to filter unreliable detections before fusion, ensuring that only consistent tracker estimates contribute to the final position estimate.

#### Ensemble Tracking

Avidan [34] introduced an ensemble tracking framework that formulates tracking as a binary classification task, combining AdaBoost-trained weak classifiers into a strong classifier to distinguish the object from the background. The tracker updates itself online by replacing outdated classifiers, enabling it to adapt to appearance changes and partial occlusions. Experiments on color, grayscale, and infrared video sequences showed robust performance, with the ensemble maintaining over 95% per-frame pixel classification accuracy under normal conditions

### Detection and Tracking of Objects in Underwater Video

Walther et al. [32] developed an automated system for underwater object detection and tracking by combining background subtraction, saliency-based attentional detection, and Kalman filter tracking. Their method effectively identified and tracked objects from underwater ROV videos, successfully detecting approximately 80–89% of annotated marine animals. However, they highlighted challenges in tracking low-contrast and elongated marine species like siphonophores.

### 2.6 Conclusion

This chapter reviewed a wide range of object tracking methodologies, from foundational techniques like Kalman filtering and optical flow to more advanced machine learning and deep learning approaches. Traditional methods, while computationally efficient and interpretable, often struggle in complex, dynamic environments. Machine learning-based trackers have improved robustness by learning representations from data but still face limitations when applied to challenging conditions such as underwater video. The OpenCV legacy tracking framework, widely used for its simplicity and real-time performance, provides a practical foundation for prototyping and experimentation.

Multi-sensor fusion techniques, including Kalman filtering and Covariance Intersection (CI), have been shown to improve tracking accuracy by aggregating information from multiple sources. CI offers a computationally efficient alternative that handles unknown correlations between sources, making it well-suited for applications where sensor interdependence cannot be modeled. The use of Mahalanobis distance adds an effective mechanism for outlier rejection, ensuring only consistent measurements are used in fusion.

In the context of underwater object tracking, prior research highlights the unique visual challenges such as low contrast, noise, and motion distortion. Benchmark studies and real-world experiments demonstrate the shortcomings of traditional and even state-of-the-art tracking algorithms in these environments. This motivates the need for adaptive, ensemble-based systems capable of maintaining accuracy under uncertainty.

In reviewing past literature, it is evident that underwater tracking remains a uniquely challenging domain due to environmental noise, appearance variation, and occlusion. Prior work has addressed some of these issues individually but often lacks a unified solution that balances robustness, adaptability, and real-time performance.

This thesis builds upon these insights by proposing a novel ensemble-based tracking framework that combines multiple OpenCV legacy trackers with Kalman filtering, Covariance Intersection, and Mahalanobis distance-based anomaly detection. While each of these components has been studied independently, their integration into a unified system for underwater fish tracking represents a new contribution. This hybrid approach aims to enhance resilience against tracker failures and environmental variability, providing a practical yet effective solution for real-world applications.

# Chapter 3: Methodology

### Introduction

This chapter outlines the methodology employed to improve tracking robustness and accuracy in underwater environments by fusing the outputs of multiple classical tracking algorithms. The proposed framework integrates six OpenCV-based trackers, each supported by a Kalman filter, and combines their outputs using statistical techniques including Mahalanobis distance-based outlier rejection and Covariance Intersection (CI). Additionally, this chapter discusses the evaluation metrics used to assess tracking performance, the dataset selected for testing, and the computational environment in which the system was implemented.

### Proposed Method

This section presents the proposed tracking framework designed to improve robustness and accuracy in underwater object tracking by fusing the outputs of multiple classical tracking algorithms. The method integrates six independent OpenCV-based trackers, each supported by a dedicated Kalman filter, with statistical outlier rejection and Covariance Intersection (CI) for state fusion. This hybrid approach enhances reliability, especially in challenging underwater conditions characterized by noise, occlusion, and dynamic lighting.

#### Multi-Tracker Ensemble Design

The tracking framework begins with an ensemble of six OpenCV legacy trackers: Boosting, MIL, KCF, MedianFlow, MOSSE, and CSRT. Each tracker is initialized with the same bounding box in the first frame and independently tracks the target fish throughout the video sequence. This ensemble provides multiple parallel hypotheses of the object’s location at each frame.

The reason for using an ensemble rather than a single tracker lies in the variability of underwater environments. Different trackers may perform better under specific visual conditions. By combining their outputs, the system increases its resilience to failures from any individual tracker.

#### Kalman Filter Integration

To improve the temporal stability of the trackers and to provide a principled representation of uncertainty, each OpenCV-based tracker is coupled with an independent Kalman filter. The Kalman filter is a recursive estimator that predicts and corrects the state of a moving object based on noisy measurements. In this system, the Kalman filter plays a dual role: it smooths the position estimates from each tracker and provides the corresponding uncertainty in the form of a covariance matrix.

Each Kalman filter models the target’s motion using a constant-velocity dynamic model. The internal state vector consists of the horizontal and vertical positions and velocities of the object, represented as

Where:

The measurement model from tracker consists of the centroid position only:

Where:

is the measurement for tracker

Each tracker’s measured position is used to correct its corresponding Kalman filter, which in turn provides a smoothed position estimate and a 2×2 covariance matrix describing the uncertainty in the position estimate.

#### Outlier Rejection using Mahalanobis Distance

Although each tracker in the ensemble is enhanced with a Kalman filter, its estimate may still be affected by errors due to occlusion, fast movement, or background interference. As a result, some trackers may produce position estimates that deviate significantly from the rest. To improve the robustness of the fusion process, these inconsistent estimates are filtered out using a statistical method based on the Mahalanobis distance.

Let denote the position estimate (the centroid) of tracker in each frame. The set of all such estimates from the valid trackers is used to compute the median position vector, denoted as ​, which serves as a robust central reference. The covariance matrix of the set of estimates is also computed to represent the spread and correlation of the data.

The Mahalanobis distance of each tracker’s estimate from the median is given by:

This metric accounts for both the distance and the statistical distribution of the data. In contrast to Euclidean distance, Mahalanobis distance normalizes for the variance and correlation among variables, making it well-suited for identifying outliers in multidimensional data.

Once the distances are computed for all trackers, the mean µ and standard deviation of the distances are calculated. A two-sigma thresholding rule is applied to retain only those trackers whose Mahalanobis distance falls within the range:

Trackers with distances outside this range are considered statistical outliers and are excluded from the fusion step. This ensures that only consistent, trustworthy position estimates contribute to the final fused output, thereby improving the stability and reliability of the tracking framework.

#### Covariance Intersection-Based Fusion

After filtering out inconsistent position estimates using Mahalanobis distance, the remaining valid estimates are fused using Covariance Intersection (CI). In the context of this study, each tracker operates independently but is influenced by the same video input, making it unsafe to assume statistical independence among trackers. CI addresses this by fusing multiple uncertain estimates without requiring knowledge of the cross-correlation between them.

Let denote the centroid position estimate from tracker , and let be the corresponding covariance matrix representing the uncertainty of that estimate. The goal of covariance intersection is to compute a fused estimate and an associated fused covariance matrix , using only the individual estimates and their covariances.

The fused covariance is computed as:

where are scalar weights assigned to each tracker such that

These weights reflect the contribution of each tracker to the final estimate and are determined through optimization. In this system, the weights are selected by minimizing the trace of the fused covariance matrix:

This objective ensures that the resulting fused estimate has the smallest total uncertainty (measured as the sum of the variances in and ), while still being statistically conservative.

Once the optimal weights are obtained, the fused estimate is computed as

This formulation guarantees that more confident (i.e., lower uncertainty) trackers have a greater influence on the result, while less confident ones contribute less. The use of the inverse covariance ​ ensures that the influence of each tracker is scaled according to its certainty.

The resulting fused position ​ serves as the best estimate of the object’s location in the current frame, and the fused covariance ​ quantifies the uncertainty of that estimate.

#### Reinitialization Strategy

To enhance robustness and reliability, the proposed tracking framework incorporates two types of reinitialization mechanisms: failure-based reinitialization and Simulated Automation with Ground-Truth policy. These mechanisms ensure that individual trackers recover from temporary failures and maintain alignment with the target, ultimately preserving the integrity of the fused estimate.

##### Failure-Based Reinitialization

During execution, each tracker is monitored for critical failure, which occurs when the .update() function fails to return a valid bounding box. Such failures typically arise from target occlusion, extreme drift, or internal instability within the tracking model. When this condition is detected, the failed tracker is automatically reinitialized to prevent its exclusion from the ensemble.

The system reinitializes the failed tracker using the current fused position estimate. This approach ensures that the recovery is internally consistent and preserves the system’s autonomy. The bounding box size used for reinitialization is calculated from the mean width and height of the currently valid trackers, ensuring continuity in scale and avoiding sudden resizing artifacts. By anchoring failed trackers to the collective fused estimate, this strategy allows for fast recovery while maintaining spatial consistency across the tracking ensemble.

A diagram of a reinitialization process



Figure 3‑1: Failure-based reinitialization logic overview

##### Simulated Automation with Ground-Truth policy

To complement the main tracking framework, a controlled mechanism called Simulated Automation with Ground-Truth is introduced. This method is distinct from failure-triggered reinitialization routines and serves a different purpose: to emulate the behavior of an external object detection module that periodically corrects tracker drift. Although this thesis does not incorporate real-time object detection, this simulated approach allows for a more structured and realistic evaluation environment.

In this simulation, each tracker is monitored to detect sustained deviations from the target. Specifically, if the predicted centroid of a tracker falls outside the ground truth bounding box for a predefined number of consecutive frames (e.g., 1, 5, or 10), the tracker is flagged for reinitialization. Once this threshold is exceeded, the tracker is reset to the ground truth bounding box location. This correction is purely evaluative and does not influence the live fusion process or fused output. Instead, it ensures that individual trackers are periodically realigned, thereby maintaining their relevance and participation in the ensemble.

Unlike reinitialization triggered by failure (e.g., based on bounding box containment or Mahalanobis-based outlier detection), this simulated method is only applied during controlled experiments. It is not an operational part of the fusion system but serves as a proxy for detection-assisted corrections in real-world deployments. The primary objective is to investigate how the ensemble behaves when such corrections are available at varying frequencies, shedding light on the trade-offs between reinitialization intervals and overall tracking stability.

### 3.3 Evaluation Metric

To comprehensively evaluate tracking performance, several key metrics are employed

#### Root Mean Squared Error (RMSE)

RMSE measures the average deviation of the predicted centroids from the ground truth across all frames. It is defined as the following:

Where is the estimated position in frame is the ground truth, and is the number of frames. Lower RMSE indicates higher tracking accuracy.

#### Euclidean Distance

For each frame, the Euclidean distance between the estimated centroid and the ground truth centroid is computed to capture per-frame accuracy

Where is the centroid of the estimated position in frame , and is the ground truth. This metric will be presented in Average distance for each tracker.

#### Intersection over Union (IoU)

IoU is used to evaluate how well the predicted bounding box overlaps with the ground truth bounding box. It is defined as

IoU is particularly useful for measuring the spatial accuracy of the bounding box in each frame.

#### Inside Target Ratio

This metric checks whether the predicted centroid falls inside the ground truth bounding box. It is expressed as a ratio over all frames

This provides an intuitive measure of spatial containment and robustness.

#### Failure and Reinitialization Count

Two metrics are used to quantify tracker reliability, failure count and reinitialization count. A failure is recorded when a tracker is unable to return a valid output via the .update() function, typically indicating a complete loss of the target. A reinitialization event, on the other hand, is triggered manually when a tracker's predicted centroid falls outside the ground truth bounding box as suggested in Simulated Automation with Ground-Truth policy. This condition ensures that drifted trackers are consistently reset and allow for an accurate evaluation of the entire framework. While failure indicates an intrinsic breakdown of the tracking model, reinitialization reflects the need for corrective intervention due to accumulated positional error. Together, these metrics provide a comprehensive view of both the internal robustness of the tracking algorithm and its dependency on external correction mechanisms.

### Data used

This study uses three underwater video sequences, each depicting a single target fish under varying environmental conditions. These videos differ in background complexity, lighting quality, turbidity, and target motion, offering a representative testbed for evaluating tracker robustness in real-world aquatic scenarios.

The videos are constructed from consecutive image frames extracted from the DeepFish dataset [1], a dataset designed to support ecological research and underwater visual analysis. Originally comprising of still images, the dataset was adapted for this research by combining sequential image together into a video to simulate realistic video-based tracking. Although the dataset contains high-quality underwater imagery, it does not provide annotated ground truth suitable for tracking. To address this, bounding box annotations were manually created by the author using the Roboflow online annotation tool. These annotations were added to each frame to enable precise, frame-by-frame comparison between tracker outputs and ground truth.

We used 3 consecutive image sequences from different environments to demonstrate the effectiveness of the tracking framework. The selected image groups are 9866\_acanthopagrus\_palmaris (447 frames), 9862\_Acanthopagrus\_palmaris (123 frames), and 7482\_F3 (239 frames). As shown in Figures 3.2 to 3.4, which will be referred to as video\_1, video\_2, and video\_3, respectively, throughout this study. These sequences were chosen to reflect a variety of environmental factors such as background clutter, turbidity, and fish behavior that influence tracking performance and provide a comprehensive basis for evaluation.

Although the number of videos used in this study is small, each video consists of a large number of consecutive frames—adding up to a total of 809 frames. This results in a substantial temporal dataset for evaluation. Therefore, while the number of videos may appear limited, the total number of frames is considerably large enough that the performance statistics are not insignificant, and the evaluation remains meaningful and representative.

The DeepFish dataset is publicly available, and further information can be found at <https://alzayats.github.io/DeepFish/>. Roboflow site is accessible through this link <https://roboflow.com/>

Example image from Video_1


Figure 3‑2: 9866\_acanthop agrus\_palmaris (Video\_1) from DeepFish dataset#1

Example image from Video_2


Figure 3‑3: 9862\_Acanthopagrus\_palmaris (Video\_2) from DeepFish dataset#2



Figure 3‑4: 7482\_F3 (Video\_3) from DeepFish dataset#3

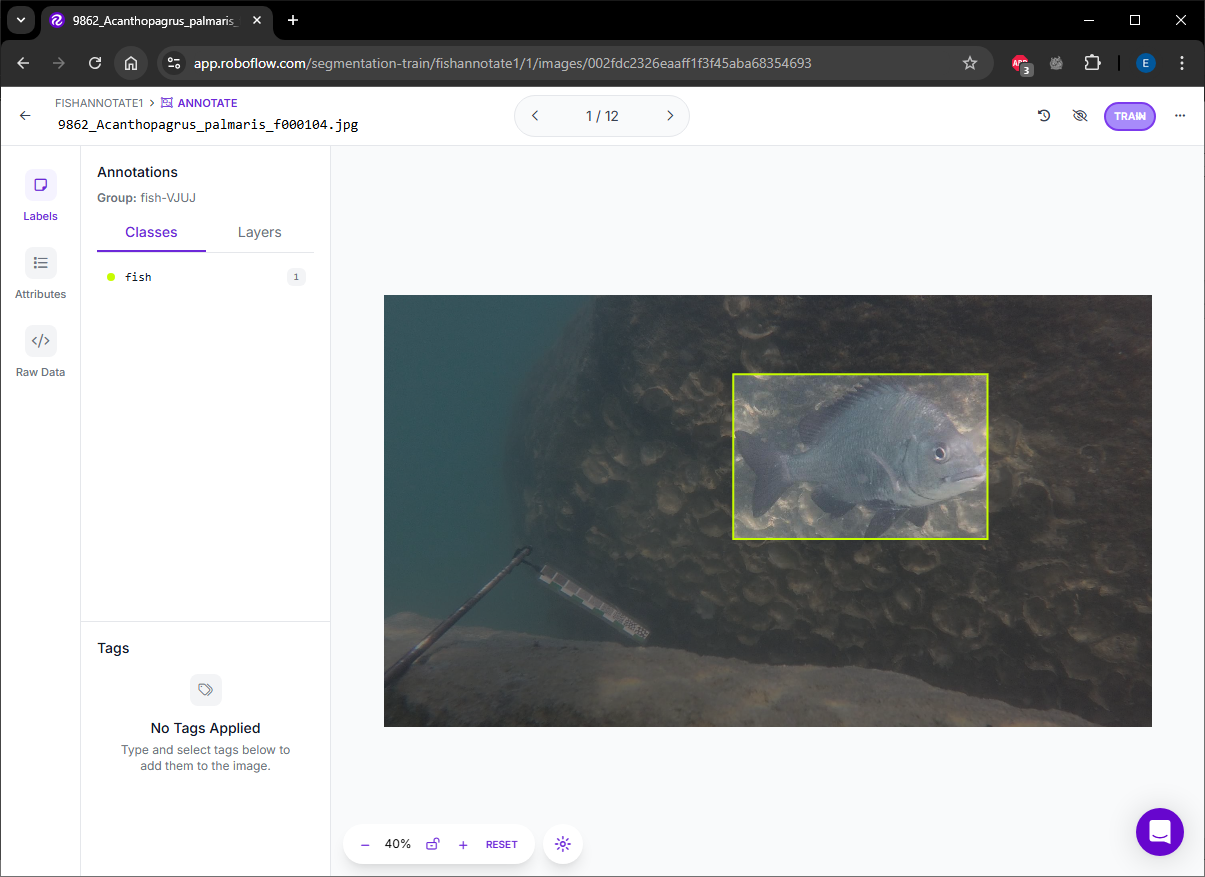


Figure 3‑5: Roboflow annotation tools

### Computational environment

All experiments were conducted on a desktop computer running Windows 10, equipped with an Intel Core i9-14900K processor and 32 GB of RAM. The implementation was developed in Python, utilizing OpenCV for object tracking, NumPy for numerical operations, and SciPy for optimization routines. Matplotlib was used for visualizing tracking performance metrics, including RMSE, Euclidean distance, and Intersection over Union (IoU) across video frames. No GPU acceleration was used, as the entire system operates on CPU-based libraries.

### Conclusion

This chapter presented a hybrid tracking methodology designed to enhance accuracy and robustness in underwater video sequences. By integrating multiple OpenCV trackers with Kalman filters, the system generates covariance associated with each tracker and further use it in covariance intersection to fuse the output. Mahalanobis distance is used to detect and discard outlier estimates, ensuring that only consistent data is passed to the fusion stage. Covariance Intersection is then applied to aggregate the remaining estimates without assuming independence between trackers, producing a final fused location with minimal uncertainty.

To maintain long-term tracking stability, the system incorporates two reinitialization strategies. Failure-based reinitialization autonomously resets a tracker when it fails to return a valid result, using the current fused position and average bounding box dimensions. In parallel, the Simulated Automation with Ground-Truth policy provides a controlled mechanism to emulate detection-assisted corrections during evaluation, enabling a structured assessment of system resilience under varying reinitialization frequencies.

Finally, the chapter outlined the evaluation metrics used to quantitatively assess system performance, described the structure and preparation of the DeepFish dataset used for testing, and reported the computational environment in which all experiments were conducted.

# Chapter 4: Experimental results

### Introduction

This chapter presents the experimental evaluation of the proposed ensemble-based tracking framework designed for underwater fish tracking. The goal is to assess its performance, reliability, and robustness under realistic aquatic conditions using annotated video sequences. Given the environmental complexity of underwater scenes—characterized by low visibility, background clutter, and erratic fish motion—the tracking task poses significant challenges for traditional single-tracker systems.

### Validation design

To validate the proposed multi-tracker fusion system, three underwater video sequences—Video\_1, Video\_2, and Video\_3, please refer to chapter 3.4 Figure 3.1-3.3—were used. These videos were selected to reflect challenging real-world conditions such as motion blur, low visibility, and background clutter, which frequently occur in underwater environments. The validation focused on measuring both tracking accuracy and robustness in a controlled, repeatable manner.

The evaluation was carried out in two operational modes—tracker and kalman. In the tracker mode, the predicted location of each tracker was taken directly from its .update() output. This raw estimate represents the tracker's immediate observation of the object’s position in each frame. In contrast, the kalman mode used the predicted output of the Kalman filter associated with each tracker. Each tracker maintained a dedicated Kalman filter, which was updated using the tracker's observation at every frame. The predicted state of the Kalman filter, after applying both correction and prediction steps, served as the final evaluated position.

For the fusion process, the framework did not rely on direct tracker outputs. Instead, it collected the state estimates and associated covariance matrices from each tracker’s Kalman filter. These estimates were filtered using a Mahalanobis distance-based outlier detection method to remove unreliable predictions. A two-sigma threshold is applied—trackers whose Mahalanobis distance falls outside this range are treated as statistical outliers and excluded from the fusion process. The remaining valid filtered states were then fused using Covariance Intersection (CI), which combined them into a single position estimate while accounting for uncertainty.

Ground truth on which the author manually annotations in every frame were provided in YOLO format for every frame and converted to absolute pixel coordinates. The centroid of the annotated bounding box served as the reference position for accuracy evaluation. Each predicted output—whether from a raw tracker, Kalman prediction, or fused estimate—was compared against this reference to calculating the final metrics.

Simulated Automation with Ground-Truth policy was incorporated not to improve fused output performance, but to simulate periodic corrections that a real-world object detection system would offer. This design ensures that evaluation results are fair and that all trackers could recover from drift, allowing a consistent comparison across the ensemble.

All outputs were evaluated on a frame-by-frame basis, and performance metrics—introduced in Section 3.3—were aggregated over the full length of each sequence. The inclusion of failure and reinitialization counts further allowed for assessment of tracker reliability and the extent of user intervention required. Together, these design elements established a comprehensive and repeatable validation protocol for evaluating the fused tracking framework under challenging real-world conditions.

### Result

#### Tracker Output Evaluation

In all three videos, the fused tracker consistently delivered the best or near-best performance in accuracy and robustness. It achieved a perfect inside ratio of 1.00 in all sequences, no tracking failures, and no reinitializations across the board. For Video\_1, the fused tracker recorded an RMSE of 53.34 and an average centroid error of 44.36, outperforming all individual trackers. In contrast, MEDIANFLOW and MOSSE exhibited 65 and 32 failures respectively, with additional user interventions, highlighting their instability.

**Video\_1 Tracker Evaluation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tracker | Average | RMSE | Inside ratio | IoU | Failed | Reinitialize Count |
| Fused | 44.36 | 53.34 | 1.00 | 0.56 | 0 | N/A |
| BOOSTING | 53.59 | 65.64 | 0.99 | 0.55 | 0 | 9 |
| CSRT | 51.25 | 63.14 | 0.99 | 0.51 | 2 | 7 |
| MEDIANFLOW | 45.68 | 55.21 | 0.85 | 0.59 | 65 | 9 |
| KCF | 61.05 | 72.68 | 0.97 | 0.50 | 3 | 8 |
| MIL | 48.50 | 56.92 | 0.99 | 0.47 | 0 | 7 |
| MOSSE | 54.53 | 70.44 | 0.92 | 0.56 | 32 | 11 |

Table 4‑1: 9866\_acanthopagrus\_palmaris Tracker Evaluation

In Video\_2, the fused tracker's performance remained dominant, achieving the lowest RMSE (36.38) and average distance (28.92). No other tracker matched this level of accuracy without requiring either failure recovery or reinitialization. For instance, MEDIANFLOW failed 14 times, and even robust performers like BOOSTING and CSRT required two reinitializations each.

**Video\_2 Tracker Evaluation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tracker | Average | RMSE | Inside ratio | IoU | Failed | Reinitialize Count |
| Fused | 28.92 | 36.38 | 1 | 0.53 | 0 | N/A |
| BOOSTING | 49.26 | 64.23 | 1 | 0.50 | 0 | 2 |
| CSRT | 50.05 | 64.06 | 1 | 0.50 | 0 | 2 |
| MEDIANFLOW | 39.02 | 45.34 | 0.81 | 0.52 | 14 | 1 |
| KCF | 34.05 | 42.89 | 0.98 | 0.51 | 3 | 2 |
| MIL | 58.28 | 82.37 | 1 | 0.51 | 0 | 1 |
| MOSSE | 61.93 | 85.02 | 0.93 | 0.50 | 6 | 3 |

Table 4‑2: 9862\_Acanthopagrus\_palmaris Tracker Evaluation

Video\_3 further validated the robustness of the fused tracker. While KCF had the lowest average distance (23.10) and RMSE (28.19), it still required three manual reinitializations, indicating fragility. The fused tracker again maintained perfect containment (inside ratio of 1.00) with zero failures or reinitializations, demonstrating that its consistent tracking is not only accurate but also sustainable over time.

**Video\_3 Tracker Evaluation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tracker | Average | RMSE | Inside ratio | IoU | Failed | Reinitialize Count |
| Fused | 32.82 | 40.34 | 1 | 0.72 | 0 | N/A |
| BOOSTING | 52.85 | 63.57 | 0.99 | 0.68 | 0 | 3 |
| CSRT | 45.58 | 59.16 | 1 | 0.69 | 0 | 3 |
| MEDIANFLOW | 46.68 | 58.72 | 0.99 | 0.68 | 0 | 5 |
| KCF | 23.10 | 28.19 | 0.99 | 0.76 | 0 | 3 |
| MIL | 30.78 | 34.34 | 1 | 0.72 | 0 | 3 |
| MOSSE | 54.98 | 68.11 | 0.98 | 0.63 | 2 | 3 |

Table 4‑3: 7482\_F3 Tracker Evaluation

#### Kalman Prediction Evaluation

The Kalman-based tracker predictions evaluations, shown in Tables 4-4, 4-5, and 4-6, reveal a similar trend. While Kalman filters provided some smoothing benefits, they also introduced greater variability in RMSE, especially when trackers failed and continued predicting without correction. For example, in Video\_2, MOSSE with Kalman filtering recorded the worst RMSE of 101.78. However, fused Kalman predictions maintained competitive RMSE across all datasets and preserved perfect inside ratios without additional errors.

The fused system consistently matches or outperforms all other trackers in every video, in every mode. Most notably, it is the only solution that did not fail or require any user intervention. This combination of high accuracy, perfect containment, and full autonomy reinforces the effectiveness of the fusion framework, particularly in challenging underwater environments. While the fused tracker itself does not require reinitialization, its consistent performance is partly supported by the recovery of individual trackers within the ensemble.

**Video\_1 Kalman Evaluation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tracker | Average | RMSE | Inside ratio | IoU | Failed | Reinitialize Count |
| Fused | 44.36 | 53.34 | 1 | 0.56 | 0 | N/A |
| BOOSTING | 56.87 | 68.71 | 0.98 | N/A | 0 | 9 |
| CSRT | 53.08 | 65.80 | 0.98 | N/A | 2 | 7 |
| MEDIANFLOW | 46.46 | 56.69 | 0.85 | N/A | 65 | 9 |
| KCF | 60.77 | 72.27 | 0.97 | N/A | 3 | 8 |
| MIL | 49.32 | 57.94 | 0.99 | N/A | 0 | 7 |
| MOSSE | 55.56 | 72.99 | 0.92 | N/A | 32 | 11 |

Table 4‑4: 9866\_acanthopagrus\_palmaris Kalman Evaluation

**Video\_2 Kalman Evaluation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tracker | Average | RMSE | Inside ratio | IoU | Failed | Reinitialize Count |
| Fused | 28.92 | 36.38 | 1 | 0.53 | 0 | N/A |
| BOOSTING | 56.92 | 75.86 | 1 | N/A | 0 | 2 |
| CSRT | 48.44 | 62.34 | 1 | N/A | 0 | 2 |
| MEDIANFLOW | 42.80 | 48.27 | 0.82 | N/A | 14 | 1 |
| KCF | 33.39 | 43.72 | 0.98 | N/A | 3 | 2 |
| MIL | 56.19 | 77.60 | 1 | N/A | 0 | 1 |
| MOSSE | 75.06 | 101.78 | 0.93 | N/A | 6 | 3 |

Table 4‑5: 9862\_Acanthopagrus\_palmaris Kalman Evaluation

**Video\_3 Kalman Evaluation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tracker | Average | RMSE | Inside ratio | IoU | Failed | Reinitialize Count |
| Fused | 32.82 | 40.34 | 1 | 0.72 | 0 | N/A |
| BOOSTING | 58.33 | 68.89 | 0.97 | N/A | 0 | 3 |
| CSRT | 52.17 | 65.42 | 0.99 | N/A | 0 | 3 |
| MEDIANFLOW | 50.13 | 61.42 | 0.99 | N/A | 0 | 5 |
| KCF | 28.16 | 33.19 | 0.99 | N/A | 0 | 3 |
| MIL | 35.90 | 39.63 | 1 | N/A | 0 | 3 |
| MOSSE | 60.86 | 74.33 | 0.99 | N/A | 2 | 3 |

Table 4‑6: 7482\_F3 Kalman Evaluation

#### Fusion Performance Summary

The superior performance of the fused tracker across all sequences comes from not only in its algorithmic design but also in its practical behavior under varying conditions. The combination of multiple diverse trackers, outlier rejection via Mahalanobis distance, and Covariance Intersection (CI) fusion enables the system to adapt to failures in its individual components without compromising the overall tracking quality.

What sets the fused tracker apart in these results is its zero-failure rate and lack of required reinitializations, even in sequences where individual trackers like MOSSE and MEDIANFLOW exhibited dozens of failures. These failures were often caused by environmental factors such as low contrast, occlusion, or target deformation—conditions that frequently occur in underwater scenes. By integrating estimates from multiple sources and rejecting inconsistent predictions, the fused tracker avoids the propagation of error that typically undermines single-model approaches.

Even high-performing trackers like KCF and CSRT, which in some cases matched or exceeded the fused tracker in one or two metrics, were still dependent on user intervention. Their reinitialization counts ranged from two to eight across different videos, indicating that their accurate outputs were not reliably sustained across full sequences. In contrast, the fused tracker’s predictions remained fully autonomous, suggesting that the collective redundancy of the fusion mechanism is more resilient than any individual tracking strategy.

In addition, the structured, Simulated Automation with Ground-Truth policy—where trackers are reset when their predicted centroid exits the ground truth bounding box—serves an important role in simulating the presence of an external object detector. While this thesis focuses solely on object tracking, the reinitialization process emulates a detection signal that realigns misaligned trackers, allowing them to recover from drift. Although the fused tracker itself never undergoes direct reinitialization, it benefits from the continuous correction of its constituent trackers, whose updated states re-enter the fusion pipeline.

Kalman filters, while useful in stabilizing predictions, did not consistently improve RMSE and often amplified error when applied to trackers that had already lost the target. In cases like MOSSE-Kalman in Video\_2, where RMSE exceeded 100, the prediction-only behavior highlighted the danger of relying solely on model-based extrapolation without fresh measurements. The fused Kalman predictions avoided these pitfalls by always grounding the estimation in multiple observations, maintaining both spatial accuracy and containment.

Altogether, the results demonstrate that accuracy alone is not sufficient for reliable object tracking—robustness and autonomy are equally essential. The fused tracker's ability to operate across all sequences with no interruptions, combined with competitive or best-in-class RMSE and average errors, confirms the practical value of the proposed framework. It provides not just a more accurate tracker, but a more dependable one—capable of sustaining long-term tracking in complex and unpredictable environments without human supervision.

### Conclusion

The experimental results presented in this chapter confirm the effectiveness and resilience of the proposed multi-tracker fusion framework. Across all three video sequences, the fused system consistently outperformed individual OpenCV trackers in terms of spatial accuracy, robustness, and autonomy. Notably, it was the only system that completed all sequences without failures or manual reinitializations.

While several individual trackers achieved good accuracy at specific moments, they frequently required external intervention or suffered from instability. The proposed framework mitigated these limitations through structured redundancy, allowing reliable estimates to dominate the fused output while suppressing inaccurate ones via Mahalanobis filtering. The Covariance Intersection mechanism further ensured that the fusion process remained statistically consistent even in the presence of unknown correlations between estimates.

The integration of a Simulated Automation with Ground-Truth policy also played a vital role. By simulating a detection system that corrected tracker drift, the framework maintained a steady pool of reliable inputs. It is important to note that this mechanism was applied solely during evaluation to simulate object detection. It does not represent a core component of the tracking framework’s runtime behavior.

Overall, the results demonstrate that the proposed system is capable of accurate and uninterrupted object tracking in underwater environments. These findings highlight the value of ensemble-based tracking with outlier rejection and uncertainty-aware fusion, setting the stage for future enhancements such as real-time reinitialization via automated detectors or integration with deep learning-based systems.

# Chapter 5: Discussion

### Introduction

This chapter discusses the implications of the experimental results introduced in Chapter 4 and system design mentioned in Chapter 3, providing an interpretation of how each component of the framework contributed to the system's overall performance. The analysis also revisits the motivations behind the use of ensemble tracking, Kalman filtering, Mahalanobis-based outlier detection, and Covariance Intersection, connecting them with observed tracking behavior. Lastly, this chapter addresses the purpose and evaluation role of the Simulated Automation with Ground-Truth policy mechanism.

### Performance of the Fused Tracker

The fused tracker consistently demonstrated high spatial accuracy, zero failures, and perfect containment across all tested sequences. Its superior performance is attributed to three synergistic factors: ensemble diversity, statistical outlier rejection, and uncertainty-aware fusion. By integrating six distinct OpenCV legacy trackers, the system was able to compensate for individual tracker failures through redundancy. Even when some trackers produced inaccurate or noisy predictions, reliable estimates from other trackers remained available for fusion. For example, as shown in Figure 5-1, the MOSSE tracker fails by drifting outside the ground truth bounding box. However, this does not negatively affect the fused tracker, which remains accurate due to contributions from the more reliable trackers. This selective integration allowed the system to adapt to variations in object appearance, motion, and environmental clutter commonly present in underwater scenes.

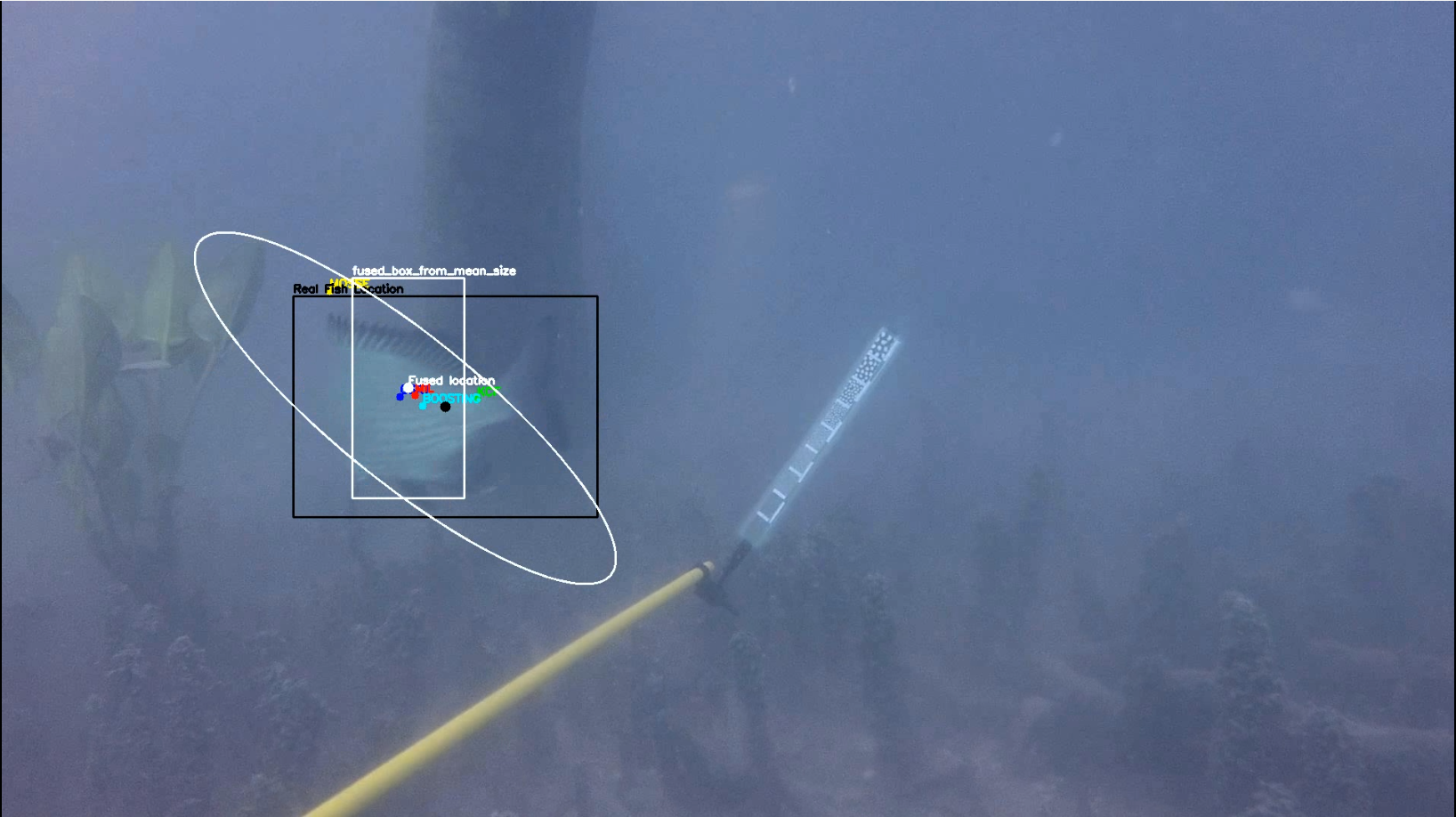


Figure 5‑1: Output Frame from Video\_1 Demonstrating Fused Tracker Stability Despite Individual Tracker Failures

The fused tracker’s stability is further supported by the RMSE-over-time graphs, particularly in Video\_1 and Video\_2. In Video\_1 (Figures A-1 and A-2), the fused tracker maintains consistently lower RMSE values compared to individual trackers in both raw and Kalman modes, reflecting strong resilience under challenging conditions. Similarly, in Video\_2 (Figures A-7 and A-8), the fused tracker shows stable and superior performance across frames, outperforming all other trackers. However, this trend does not continue in Video\_3 (Figures A-13 and A-14), where the fused tracker performs near the average, with some individual trackers achieving better RMSE. This suggests that in scenarios where the target is easier to track—due to clearer visual features or reduced environmental clutter—the advantage of the fusion framework may diminish, and the added complexity may not lead to significant performance gains.

This trend also applies to other performance metrics, as illustrated by the graphs in the Appendix. The fused tracker generally outperforms individual trackers across metrics such as Euclidean distance and Intersection over Union (IoU). While the upper bound of each metric is inherently limited by the best-performing individual tracker at any given moment, the fused tracker demonstrates superior consistency over time. By leveraging contributions from multiple trackers and filtering out outliers, the fused output achieves more stable and reliable performance in the long run—surpassing what any single tracker can maintain across an entire sequence.

### Role of Kalman Filtering

While Kalman filters are commonly used for motion prediction and smoothing, in this study, their primary role was to extract uncertainty information (covariance matrices) for each tracker. The OpenCV Python implementation does not expose internal covariance from its legacy trackers, which makes it challenging to reason about the reliability of their predictions. By coupling each tracker with a dedicated Kalman filter, the system gained access to a covariance estimate for every frame. This enabled informed fusion using Covariance Intersection.

Although Kalman filtering provided temporal smoothing benefits, the experiments showed that it could also introduce drift when a tracker failed but continued predicting based on outdated internal state. This behavior was particularly evident in high-failure cases like MOSSE in Video\_2. Therefore, Kalman outputs should not be interpreted as more accurate by default. Their utility lies in contributing uncertainty-aware estimates rather than necessarily improving spatial precision.

### Role of Outliner filtering utilizing Mahalanobis Distance

Mahalanobis distance filtering further enhanced this process by removing inconsistent estimates before fusion. Instead of treating all trackers equally, the system retained only those whose predictions were statistically consistent with the group. This step was particularly effective in reducing the impact of erratic behavior from unreliable trackers like MOSSE or MEDIANFLOW in challenging frames. By applying a two-sigma rule, the framework ensured robust ensemble consistency without being overly restrictive.

To explore the sensitivity of the filter, a 1.5-sigma threshold was also evaluated. While this tighter bound excluded more trackers and potentially further reduced the influence of outliers, the resulting tracking performance was slightly worse than with the 2-sigma configuration. This is because the Mahalanobis filter operates relative to the statistical center of the tracker ensemble—essentially, the mean of all active predictions—not the actual ground truth. Therefore, removing more estimates does not guarantee convergence toward the true object location. In fact, over-filtering can degrade accuracy by discarding useful trackers that deviate slightly from the ensemble center but are still closer to the true target.

This result underscores a subtle but important distinction that the ensemble center is not the same as the ground-truth fish location. It is merely the centroid of the tracker predictions after accounting for covariance. As a result, while 1.5-sigma enforces stricter filtering, it may exclude trackers that are correct but simply offset from the mean of the group. The 2-sigma rule thus provides a better balance between removing genuine outliers and retaining diversity within the ensemble—especially when the true object location is not well-aligned with the ensemble mean due to drift, occlusion, or tracker failures.

**1.5 Sigma rule Video\_1**

|  |  |  |  |
| --- | --- | --- | --- |
| Tracker | # of filtered out | Average | RMSE |
| CSRT | 71 | N/A | N/A |
| BOOSTING | 14 | N/A | N/A |
| KCF | 34 | N/A | N/A |
| MIL | 77 | N/A | N/A |
| MEDIANFLOW | 37 | N/A | N/A |
| MOSSE | 5 | N/A | N/A |
| Fused | N/A | 44.48 | 53.68 |

Table 5‑1: Number of filtered trackers and performance of the fused tracker on Video\_1 with 1.5-Sigma rule in Kalman setting

**2 Sigma rule Video\_1**

|  |  |  |  |
| --- | --- | --- | --- |
| Tracker | # of filtered out | Average | RMSE |
| CSRT | 1 | N/A | N/A |
| MIL | 4 | N/A | N/A |
| MEDIANFLOW | 5 | N/A | N/A |
| Fused | N/A | 44.36 | 53.34 |

Table 5‑2: Number of filtered out trackers and performance of the fused tracker on Video\_1 with 2-Sigma rule in Kalman setting

### Role of Covariance Intersection (CI)

Covariance Intersection (CI) provided the final step of fusing valid estimates without assuming independence between them. This was crucial since the trackers operated on the same input data and could not be assumed to generate statistically independent outputs. CI used the uncertainty (covariance) of each tracker to determine its influence on the final estimate. Trackers with low positional variance contributed more to the final position, while high-uncertainty trackers had a diminished influence.

Although Covariance Intersection (CI) is designed to assign greater influence on trackers with lower uncertainty, in this study, the Kalman filters associated with each tracker produced nearly identical covariance matrices. As a result, the CI optimization consistently yielded uniform weights across all trackers, effectively averaging the filtered predictions rather than performing a confidence-weighted fusion. This outcome can be attributed to the uniform configuration of all Kalman filters—each initialized with the same process and measurement noise parameters—and the fact that they were updated at every frame whenever a tracker produced an output, without evaluating whether the output was a false positive. This caused the filters to become more confident than they should be. Additionally, since all filters operated under similar conditions, their internal estimates evolved in a highly consistent manner. While this uniformity contributes to the overall stability of the system, it also limits the variation in uncertainty required for CI to assign differentiated weights. Future work may benefit from introducing tracker-specific noise parameters— based on each tracker's observed accuracy— or by adopting adaptive uncertainty modeling, such as updating each Kalman filter only if the tracker is retained after outlier rejection. This could help reflect the varying reliability of individual trackers.



### Purpose of Simulated Automation with Ground-Truth policy

The Simulated Automation with Ground-Truth policy policy played a critical role in maintaining the quality of inputs fed into the fusion process. However, it is important to emphasize that this mechanism was not designed to improve the fused output directly. Instead, it served as a controlled simulation of an external object detection system.

In real-world object tracking applications, trackers are often supported by detection modules that periodically reset their state. Since this study focused exclusively on tracking, the Simulated Automation with Ground-Truth policy fulfilled that function. Trackers whose predicted centroid remained outside the ground truth bounding box for a predefined duration (1, 5, or 10 frames) were manually realigned using ground truth annotations. This ensured that drifted trackers could rejoin the ensemble and continue contributing valid data.

The fused tracker never underwent direct reinitialization. Its consistent performance was enabled by the recovery of its constituent trackers, which were periodically corrected through this rule. Thus, reinitialization served as an evaluative function: it allowed the system to generate reliable results over long sequences, enabling a fair and consistent comparison of tracking behavior.

### Validation of Simulated Automation with Ground-Truth Policy and Its Influence on Fused Output

To further validate the importance of periodic external correction within the proposed framework, an experiment was conducted on Video\_1 using three variants of the Simulated Automation with Ground-Truth policy. In this experiment, the tracker was manually reinitialized if its predicted centroid remained outside the ground truth bounding box for 1, 5, or 10 consecutive frames. The goal was to assess how the frequency of reinitialization affects tracking performance, particularly in relation to the fused output.

As shown in Table 5-3 to 5-5, the fused tracker consistently demonstrated better performance with more frequent reinitialization. When the threshold was set to 1 frame, the fused tracker achieved the lowest RMSE of 53.34 and a perfect inside ratio of 1.00. Under a 5-frame threshold, performance slightly declined to an RMSE of 55.45 and an inside ratio of 0.98. When reinitialization was delayed until 10 frames of persistent failure, the fused tracker's RMSE rose substantially to 71.20, with the inside ratio dropping to 0.96.

This trend was consistent across individual trackers. For instance, CSRT’s RMSE increased from 55.77 in the 5-frame case to 87.55 when reinitialization was only triggered after 10 frames. MOSSE, which is known for its high speed but lower reliability, exhibited a sharp performance drop as the delay in reinitialization increased. These results confirm that frequent reinitialization effectively prevents long-term drift and maintains spatial accuracy across trackers.

Importantly, while the fused estimate itself is never manually reinitialized, its success is enabled by the reinitialization applied to its constituent trackers. These reinitializations help restore alignment and ensure that useful tracking data remains available for the fusion process. The system does not require all trackers to always function perfectly but rather thrives by identifying and relying on those that are accurate in any given frame. When some trackers fail, others continue to contribute valid data, and once failed trackers are reinitialized, they can rejoin the fusion process seamlessly.

This mechanism promotes both robustness and efficiency. It allows trackers to fail and recover without derailing the entire system and ensures that computational resources are used effectively—only trustworthy predictions are fused. As a result, the system avoids error accumulation and maintains high performance even in dynamic or visually challenging environments such as underwater video.

Overall, this validating experiment confirms that the proposed fusion framework is not only accurate but also structurally resilient. It adapts to fluctuations in tracker quality, tolerates temporary degradation, and leverages timely corrections to sustain long-term tracking performance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Video\_1 tracker Evaluation 10 frame** | | | | |
| Tracker | Average | RMSE | Inside ratio | IoU |
| Fused | 53.90 | 71.20 | 0.96 | 0.55 |
| BOOSTING | 63.08 | 83.80 | 0.94 | 0.54 |
| CSRT | 55.58 | 87.55 | 0.95 | 0.59 |
| MEDIANFLOW | 54.26 | 69.67 | 0.84 | 0.59 |
| KCF | 64.01 | 83.12 | 0.92 | 0.52 |
| MIL | 55.64 | 72.20 | 0.97 | 0.53 |
| MOSSE | 78.72 | 104.73 | 0.83 | 0.49 |

Table 5‑3: Video\_1 Result with 10-frames rule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Video\_1 tracker Evaluation 5 frame** | | | | |
| Tracker | Average | RMSE | Inside ratio | IoU |
| Fused | 43.87 | 55.45 | 0.98 | 0.57 |
| BOOSTING | 54.04 | 67.86 | 0.96 | 0.55 |
| CSRT | 41.93 | 55.77 | 0.98 | 0.57 |
| MEDIANFLOW | 41.02 | 51.29 | 0.84 | 0.61 |
| KCF | 57.52 | 70.53 | 0.97 | 0.54 |
| MIL | 42.81 | 55.25 | 0.93 | 0.58 |
| MOSSE | 68.59 | 87.68 | 0.90 | 0.53 |

Table 5‑4: Video\_1 Result with 5-frames rule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Video\_1 tracker Evaluation 1 frame** | | | | |
| Tracker | Average | RMSE | Inside ratio | IoU |
| Fused | 44.36 | 53.34 | 1.00 | 0.56 |
| BOOSTING | 53.59 | 65.64 | 0.99 | 0.55 |
| CSRT | 51.25 | 63.14 | 0.99 | 0.51 |
| MEDIANFLOW | 45.68 | 55.21 | 0.85 | 0.59 |
| KCF | 61.05 | 72.68 | 0.97 | 0.50 |
| MIL | 48.50 | 56.92 | 0.99 | 0.47 |
| MOSSE | 54.53 | 70.44 | 0.92 | 0.56 |

Table 5‑5: Video\_1 Result with 1-frames rule

### Implications for Real-World Deployment

The system's ability to function without any direct intervention at the fused output level suggests strong potential for deployment in autonomous monitoring systems. Marine biologists and ecological researchers could use this tracking framework to study fish behavior over extended periods without constant manual supervision. However, this deployment would still require a robust object detector to handle reinitializations automatically, replacing the manual resets used during evaluation.

Future versions of this system could integrate a deep learning-based object detector to enable real-time detection-triggered reinitialization. Such integration would fully automate the pipeline, transitioning it from a semi-automated research framework to a practical tool for real-world use.

### Conclusion

This chapter interpreted the experimental results of the proposed tracking framework. The fused tracker outperformed all individual trackers by leveraging ensemble diversity, statistical filtering via Mahalanobis distance, and covariance-aware fusion. Kalman filters enabled access to uncertainty metrics necessary for Covariance Intersection, rather than improving accuracy alone. Reinitialization, while enhancing consistency, was not a performance booster but a detection simulation mechanism, ensuring fair and stable evaluation. Overall, the system’s robust, adaptive behavior affirms the value of ensemble-based tracking in challenging underwater environments.

# Chapter 6: Conclusion and Future Work

### Conclusion

This thesis presented a semi-autonomous ensemble-based tracking framework aimed at enhancing the accuracy and reliability of underwater single-fish tracking. By integrating multiple OpenCV legacy trackers with individual Kalman filters, applying Mahalanobis distance-based outlier rejection, and fusing predictions using Covariance Intersection (CI), the system demonstrated significant improvements in stability and robustness over traditional single-tracker methods.

Experiments conducted on three underwater video sequences from the DeepFish dataset confirmed the effectiveness of the proposed approach. The fused tracker consistently achieved the highest accuracy, maintained perfect containment ratios, and required no reinitialization, even under challenging visual conditions such as background clutter, occlusion, and rapid fish motion. In contrast, individual trackers often exhibited failures or required external correction.

A key insight from this study is that, while CI is designed to perform confidence-weighted fusion, the Kalman filters used in this implementation generated nearly identical covariance matrices. This led CI to assign uniform weights across all trackers, resulting in a balanced rather than differentiated fusion. This outcome highlights the importance of incorporating tracker-specific uncertainty modeling to fully leverage CI’s potential.

In addition, the system's rule-based reinitialization strategy played an important role in maintaining tracker alignment, simulating how an object detection system might intervene in real-world applications. Although the fused estimate never required reinitialization, its sustained performance was enabled by the ability of individual trackers to recover and rejoin the fusion process.

Overall, this framework offers a practical and robust solution for underwater tracking tasks and lays the groundwork for more adaptive and intelligent tracking systems in marine biology and ecological monitoring.

### Future Work

#### Adaptive Kalman Filter Configuration

Future versions of the framework could benefit from adaptive noise modeling. By adjusting each tracker's process and measurement noise based on its historical accuracy or reliability, the resulting covariance matrices may better reflect the true uncertainty of each estimate, allowing CI to perform more meaningful confidence-weighted fusion.

In addition, Kalman filters should only be updated when a tracker's prediction is validated as non-anomalous, for example, by passing Mahalanobis distance-based outlier detection. By avoiding updates with potentially faulty measurements, the filters can maintain a more accurate internal state and avoid becoming overconfident due to consistently updating with poor or misleading data. This selective update mechanism would enhance the trustworthiness of the fusion inputs and prevent error accumulation from unreliable trackers.

Together, these modifications would make the fusion process more reflective of real uncertainty and better aligned with each tracker's dynamic behavior.

#### Integration with Deep Learning-Based Detectors

Although this study focused solely on tracking, integrating a lightweight object detector could allow for true tracking-by-detection pipelines. Detection results could periodically reinitialize the ensemble, improving long-term robustness and reducing the reliance on manual initialization or rule-based resets.

#### Expansion to Multi-Object Tracking

While this work focuses on single-fish tracking, the underlying fusion and outlier rejection mechanisms can be extended to support multiple objects. Handling object association, identity preservation, and real-time reinitialization will be key challenges in this area.

#### Dataset Expansion and Diversity

Although the current study utilizes 809 frames across three videos, which is sufficient to ensure statistically non-insignificant evaluation, the diversity of environmental conditions and fish behaviors remains limited. In future work, the framework will be extended to include more videos and species, covering a broader range of underwater conditions such as varying turbidity, lighting, and background clutter. Expanding the dataset will improve generalizability and strengthen the empirical foundation of the results.

#### Investigation of Tracker Variants and Scalability

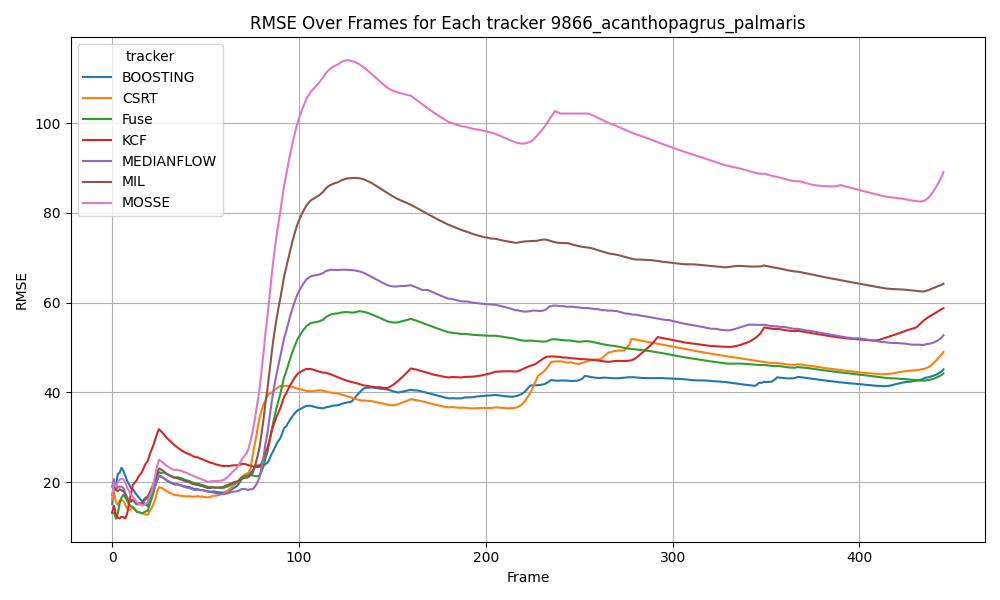
The current tracker set is limited to six OpenCV legacy trackers. Future research will explore different combinations of trackers—including both traditional and modern algorithms—to evaluate how tracker diversity and selection influence fusion quality and robustness.

Additionally, the scalability of the framework will be investigated. While increasing the number of trackers may offer marginal improvements in robustness, it may also significantly impact time complexity. A deeper analysis is planned to understand the trade-offs between performance gain and computational cost—for example, testing with 10, 20, or even 100 trackers to determine the point of diminishing returns.

# Bibliography

|  |  |
| --- | --- |
| [1] | OpenCV., "Optical flow tutorial.," [Online]. Available: https://docs.opencv.org/5.x/d4/dee/tutorial\_optical\_flow.html. [Accessed 1 9 2024]. |
| [2] | G. Welch and G. Bishop, "An Introduction to the Kalman Filter," in *SIGGRAPH 2001 Course Notes*, ACM, 2001. |
| [3] | S. Manzoor, K.-H. Sung, Y. Zhang, Y.-C. An and . T.-Y. Kuc, "Qualitative Analysis of Single Object and Multi Object Tracking Models," in *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, 2022, pp. 1539-1545. |
| [4] | A. Lukežič, T. Vojíř, L. Č. Zajc, J. Matas and M. Kristan, "Discriminative Correlation Filter Tracker with Channel and Spatial Reliability," *International Journal of Computer Vision,* vol. 126, no. 7, pp. 671-688, 2018. |
| [5] | D. S. Bolme, J. R. Beveridge, B. A. Draper and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2544-2550. |
| [6] | Z. Kalal, K. Mikolajczyk and J. Matas, "Forward-Backward Error: Automatic Detection of Tracking Failures," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 2756-2759. |
| [7] | H. Grabner, M. Grabner and H. Bischof, "Real-Time Tracking via On-line Boosting," *Proceedings of British Machine Vision Conference (BMVC),* vol. 1, pp. 47-56, 2006. |
| [8] | B. Babenko, M.-H. Yang and S. Belongie, "Visual tracking with online Multiple Instance Learning," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 983-990. |
| [9] | J. F. Henriques, R. Caseiro, P. Martins and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, no. 3, pp. 583-596, 2015. |
| [10] | D. Huang, M. Yang, J. Duan, S. Yu and Z. Liu, "Siamese Network Tracking Based on Feature Enhancement," *IEEE Access,* vol. 11, pp. 37705-37713, 2023. |
| [11] | LearnOpenCV. , "The complete guide to object tracking – OpenCV, DeepSort, FairMOT.," [Online]. Available: https://learnopencv.com/the-complete-guide-to-object-tracking-in-computer-vision/. [Accessed 1 10 2023]. |
| [12] | OpenCV, "Legacy Tracking API," [Online]. Available: https://docs.opencv.org/4.8.0/dc/d6b/group\_\_tracking\_\_legacy.html. [Accessed 01 09 2023]. |
| [13] | Pierian Training , "Kalman Filter OpenCV Python Example," 26 5 2023. [Online]. Available: https://pieriantraining.com/kalman-filter-opencv-python-example/. [Accessed 1 10 2023]. |
| [14] | OpenCV. , "KalmanFilter class reference," [Online]. Available: https://docs.opencv.org/3.4/dd/d6a/classcv\_1\_1KalmanFilter.html. [Accessed 21 2 2024]. |
| [15] | H. Ghorbani, "MAHALANOBIS DISTANCE AND ITS APPLICATION FOR DETECTING MULTIVARIATE OUTLIERS," *Facta Universitatis Series Mathematics and Informatics,* vol. 34, p. 583, 2019. |
| [16] | Stone Soup, "Multi-sensor fusion: Covariance intersection using tracks as measurements.," [Online]. Available: https://stonesoup.readthedocs.io/en/stable/auto\_examples/Track2Track\_Fusion\_Example.html. [Accessed 20 4 2024]. |
| [17] | R. Forsling, B. Noack and G. Hendeby, "A Quarter Century of Covariance Intersection: Correlations Still Unknown? [Lecture Notes]," *IEEE Control Systems Magazine,* vol. 44, no. 2, pp. 81-105, 2024. |
| [18] | A. Saleh, I. H. Laradji, D. A. Konovalov, M. Bradley, D. Vazquez and M. Sheaves, "A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis," *Scientific Reports,* vol. 10, no. 14671, 2020. |
| [19] | Z.-Q. WEN and Z.-X. CAI, "Mean Shift Algorithm and its Application in Tracking of Objects," in *2006 International Conference on Machine Learning and Cybernetics*, 2006, pp. 4024-4028. |
| [20] | OpenCV., "Meanshift and Camshift.," [Online]. Available: https://docs.opencv.org/5.x/d7/d00/tutorial\_meanshift.html. [Accessed 1 11 2024]. |
| [21] | J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788. |
| [22] | S. Kanimozhi , G. Gayathri and T. Mala , "Multiple Real-time object identification using Single shot Multi-Box detection," in *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, 2019, pp. 1-5. |
| [23] | broutonlab., "A complete review of the OpenCV object tracking algorithms.," [Online]. Available: https://broutonlab.com/blog/opencv-object-tracking/. [Accessed 1 10 2023]. |
| [24] | N. Dardagan, A. Brđanin, D. Džigal and A. Akagic, "Multiple Object Trackers in OpenCV: A Benchmark," in *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, 2021, pp. 1-6. |
| [25] | A. A. Levin, D. D. Klimov, A. A. Nechunaev, L. S. Prokhorenko, D. S. Mishchenkov, A. G. Nosova, D. A. Astakhov, Y. V. Poduraev and D. N. Panchenkov, "Assessment of experimental OpenCV tracking algorithms for ultrasound videos," *Scientific Reports,* vol. 13, no. 1, p. 6765, 2023. |
| [26] | A. Yilmaz, O. Javed and M. Shah, Object tracking: A survey, vol. 38, New York, NY: Association for Computing Machinery, 2006, p. 13–es. |
| [27] | Y. Wu, J. Lim and M.-H. Yang, "Object Tracking Benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, no. 9, pp. 1834-1848, 2015. |
| [28] | A. Al Muksit, F. Hasan, F. H. M. Bhuiyan Emon, M. Rakibul Haque, A. Reza Anwary and S. Shatabda, "YOLO-Fish: A robust fish detection model to detect fish in realistic underwater environment," *Ecological Informatics,* vol. 72, p. 101847, 2022. |
| [29] | K. Panetta, L. Kezebou, V. Oludare and S. Agaian, "Comprehensive Underwater Object Tracking Benchmark Dataset and Underwater Image Enhancement With GAN," *IEEE Journal of Oceanic Engineering,* vol. 47, no. 1, pp. 59-75, 2022. |
| [30] | R. Anitha, S. Renuka and A. Abudhahir, "Multi sensor data fusion algorithms for target tracking using multiple measurements," in *2013 IEEE International Conference on Computational Intelligence and Computing Research*, 2013, pp. 1-4. |
| [31] | Y. Jiang and J. Xiao, "Target tracking based on a multi-sensor covariance intersection fusion Kalman filter," *Engineering Review,* vol. 34, no. 1, pp. 47-54, 2014. |
| [32] | D. Walther, D. R. Edgington and C. Koch, "Detection and tracking of objects in underwater video," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1, 2004, pp. I-I. |
| [33] | B. Wei, Y. Wang and X. He, "Confidence map based KCF object tracking algorithm," in *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2019, pp. 2187-2192. |
| [34] | N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886-893 vol. 1. |
| [35] | L. Kezebou, V. Oludare, K. Panetta and S. S. Agaian, "Underwater Object Tracking Benchmark and Dataset," in *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*, 2019, pp. 1-6. |
| [36] | C. Spampinato, S. Palazzo, D. Giordano, I. Kavasidis, F.-P. Lin and Y.-T. Lin, "Covariance based Fish Tracking in Real-life Underwater Environment.," in *VISAPP (2)*, 2012, pp. 409--414. |
| [37] | S. Avidan, "Ensemble Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 29, no. 2, pp. 261-271, 2007. |

## Appendix

Appendix A: Graph demonstrates the output of each metric over time

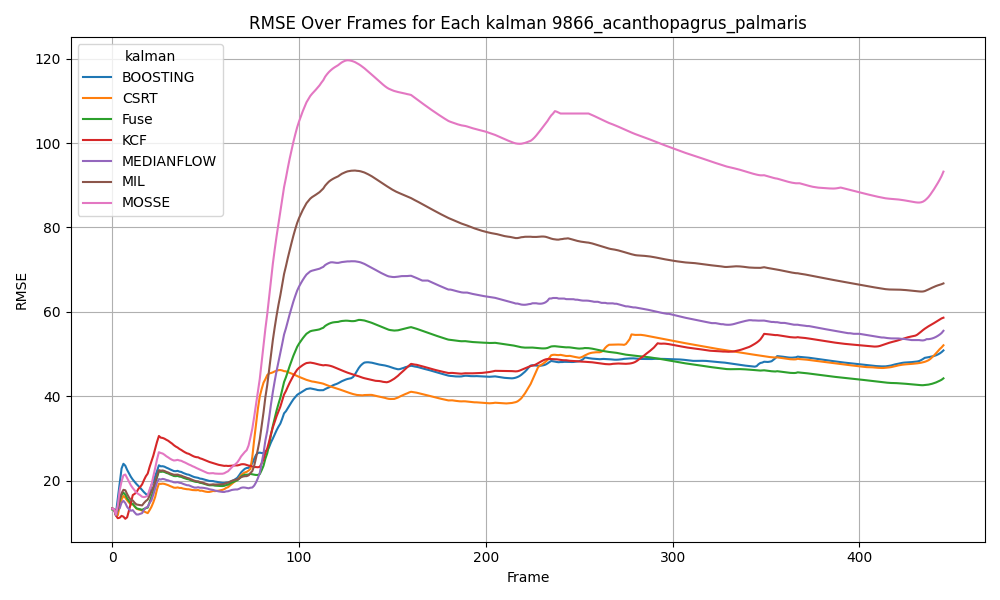
Figure A‑1: Video\_1 RMSE over time for Tracker

Figure A-2: Video\_1 RMSE over time for Kalman

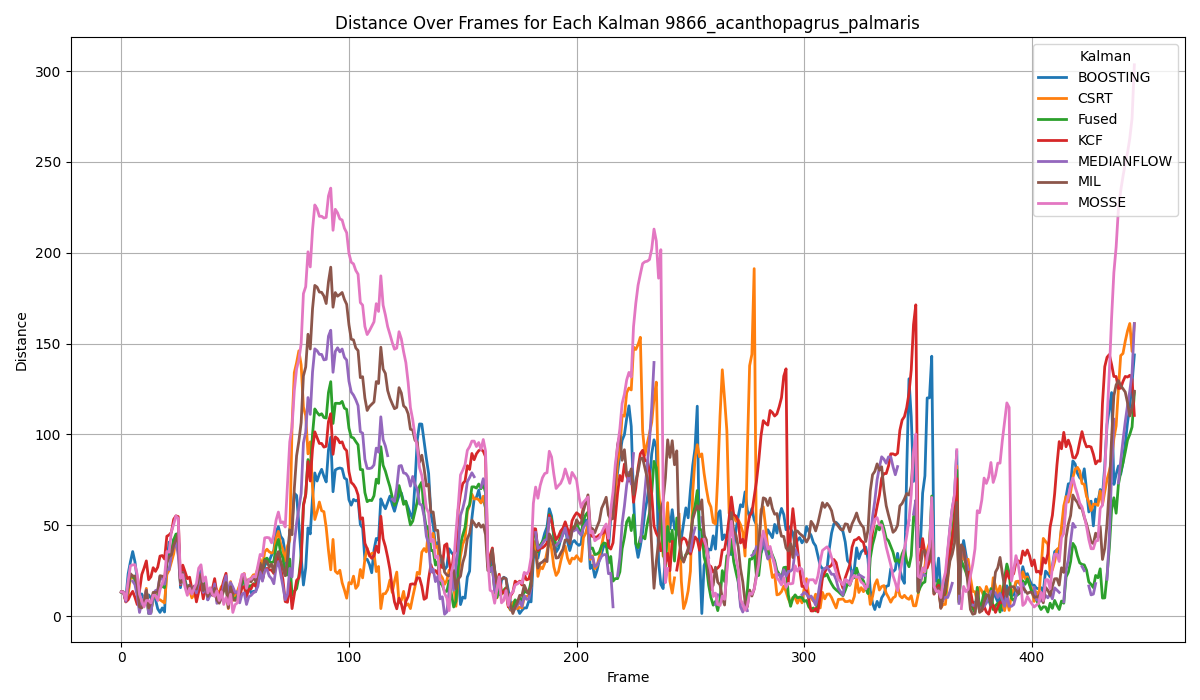


Figure A‑3: Video\_1 Euclidean distance over time for Kalman

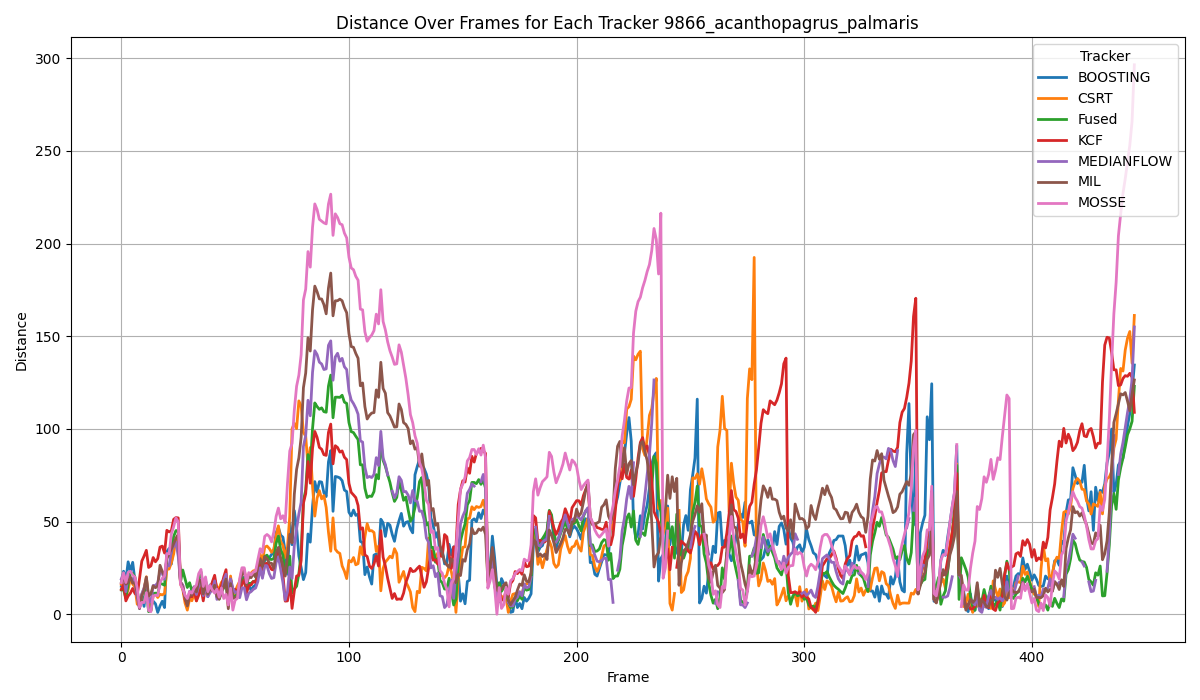


Figure A-4: Video\_1 Euclidean distance over time for Tracker

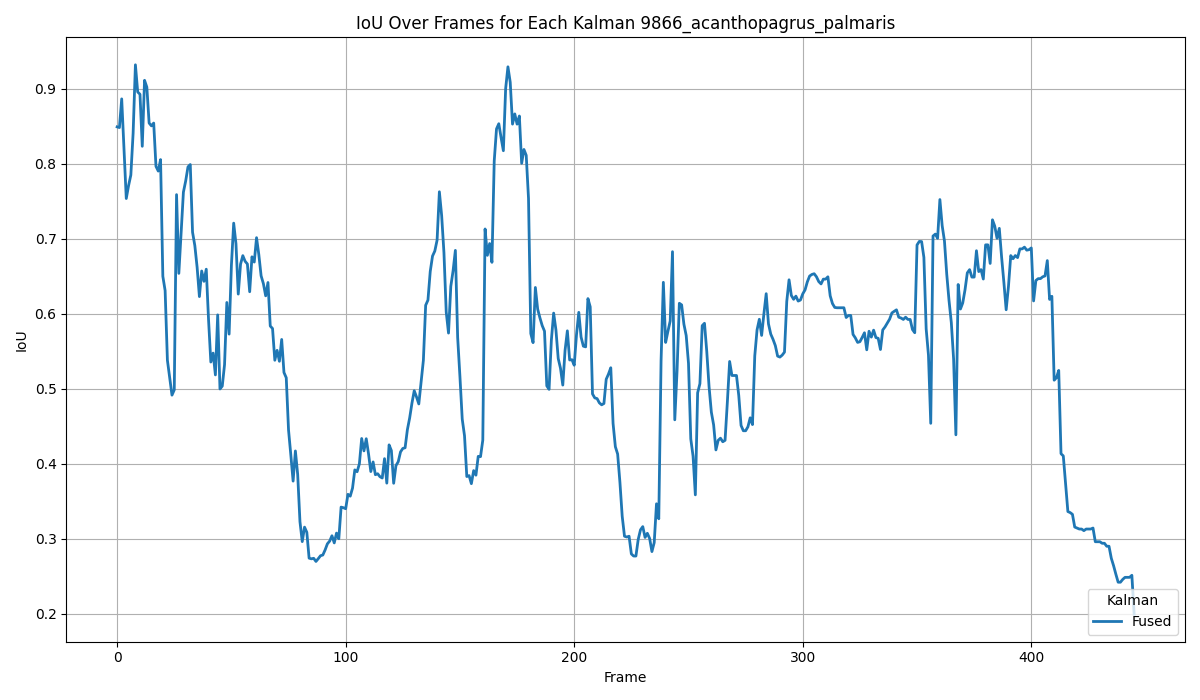


Figure A-5: Video\_1 IoU over time for Kalman

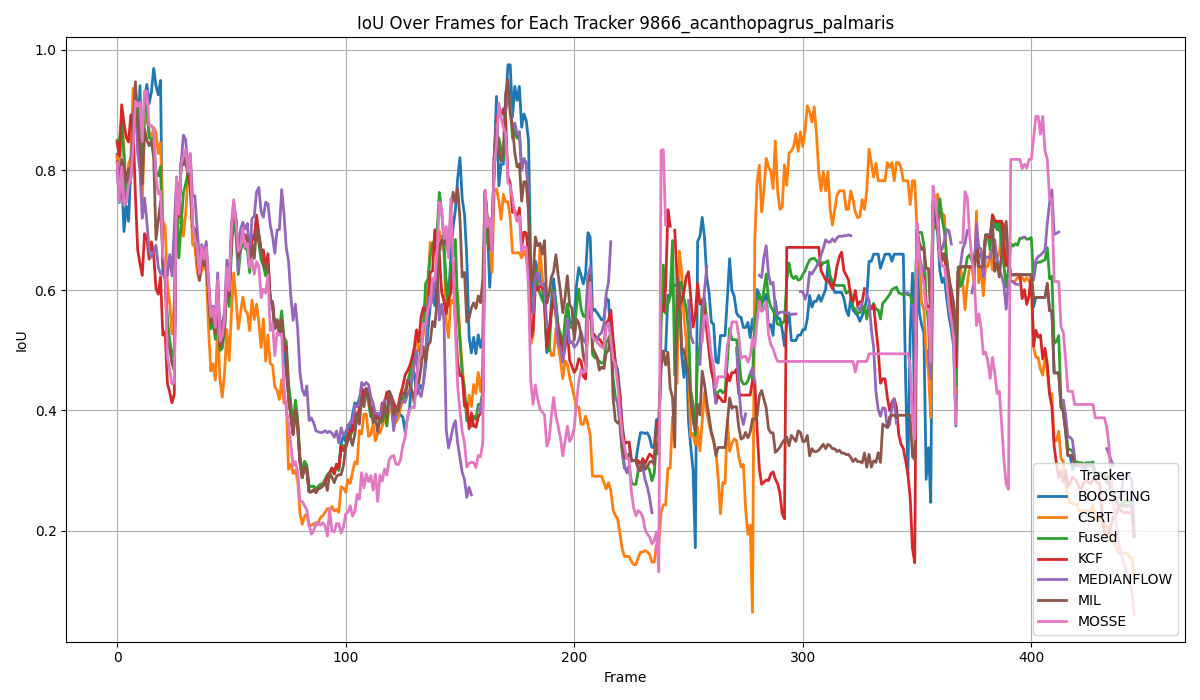


Figure A‑6: Video\_1 IoU over time for Tracker

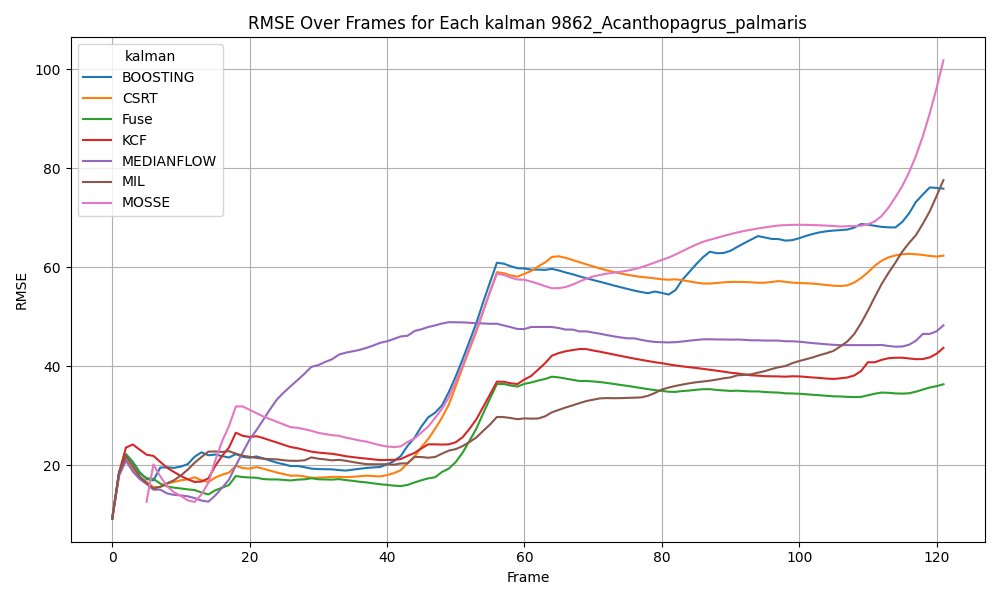


Figure A‑7: Video\_2 RMSE over time for Kalman

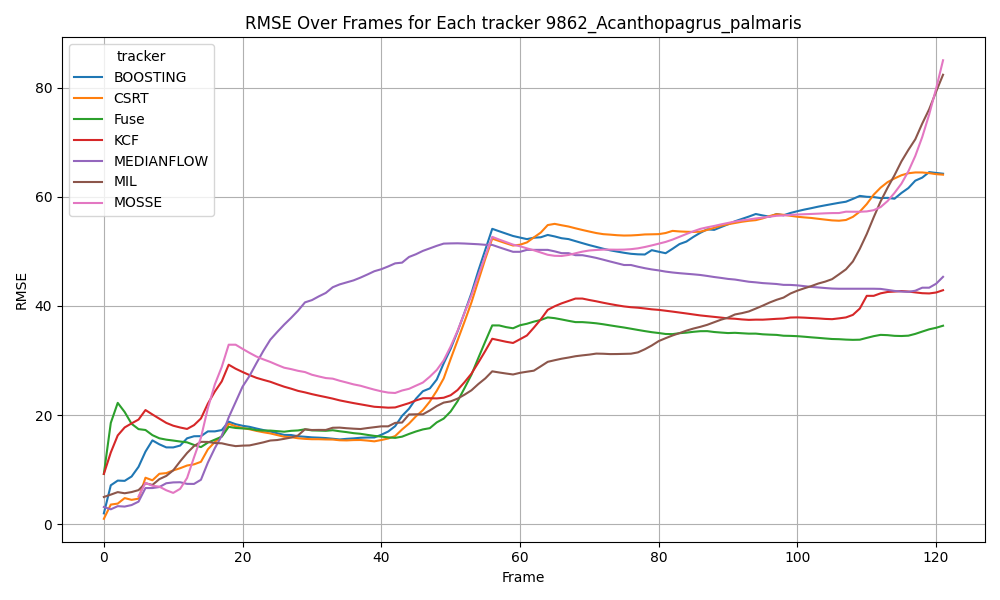


Figure A‑8: Video\_2 RMSE over time for Tracker

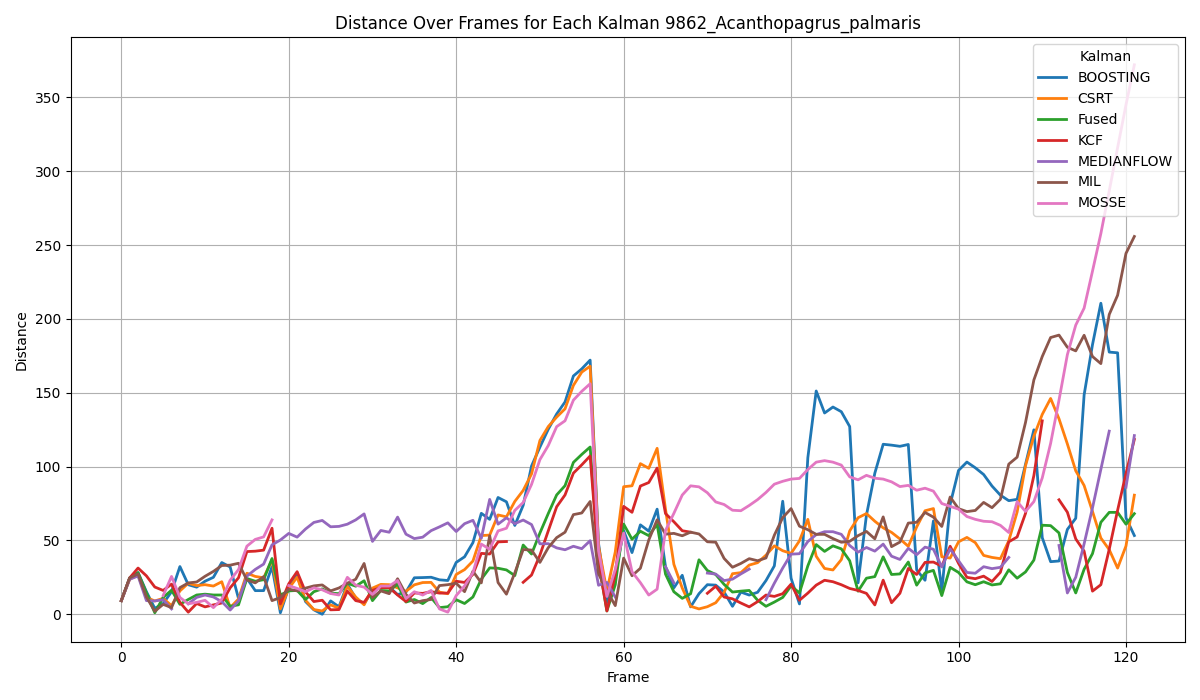


Figure A‑9: Video\_2 Euclidean distance over time for Kalman

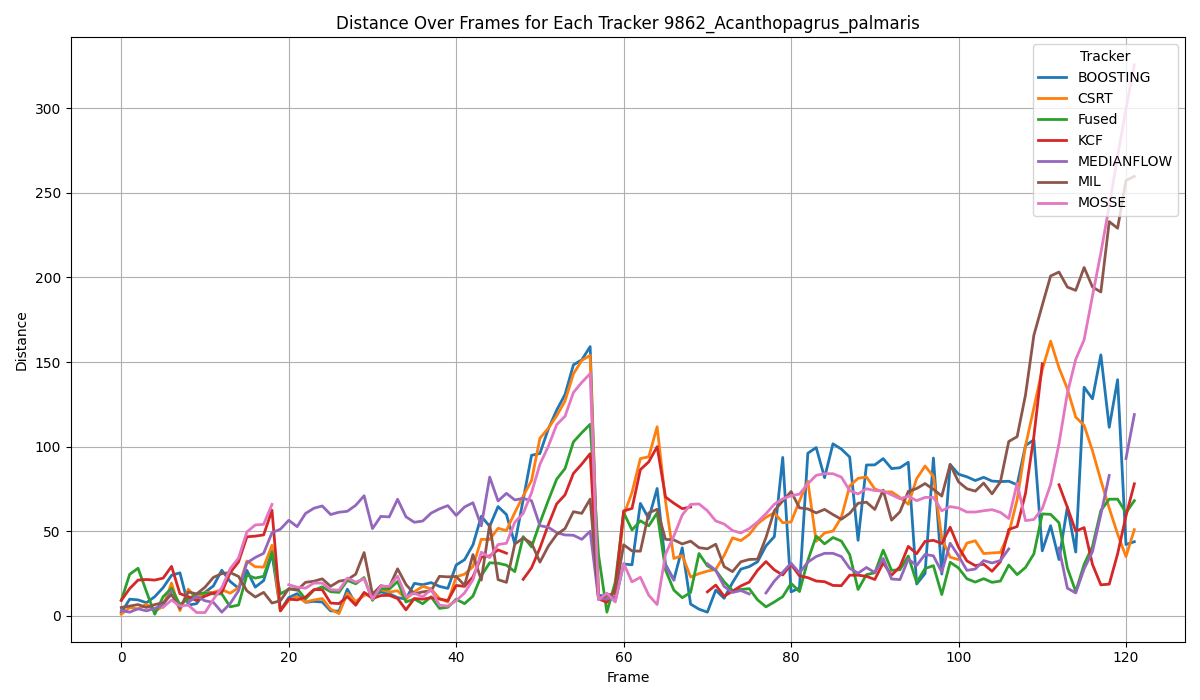


Figure A-10: Video\_2 Euclidean distance over time for Tracker

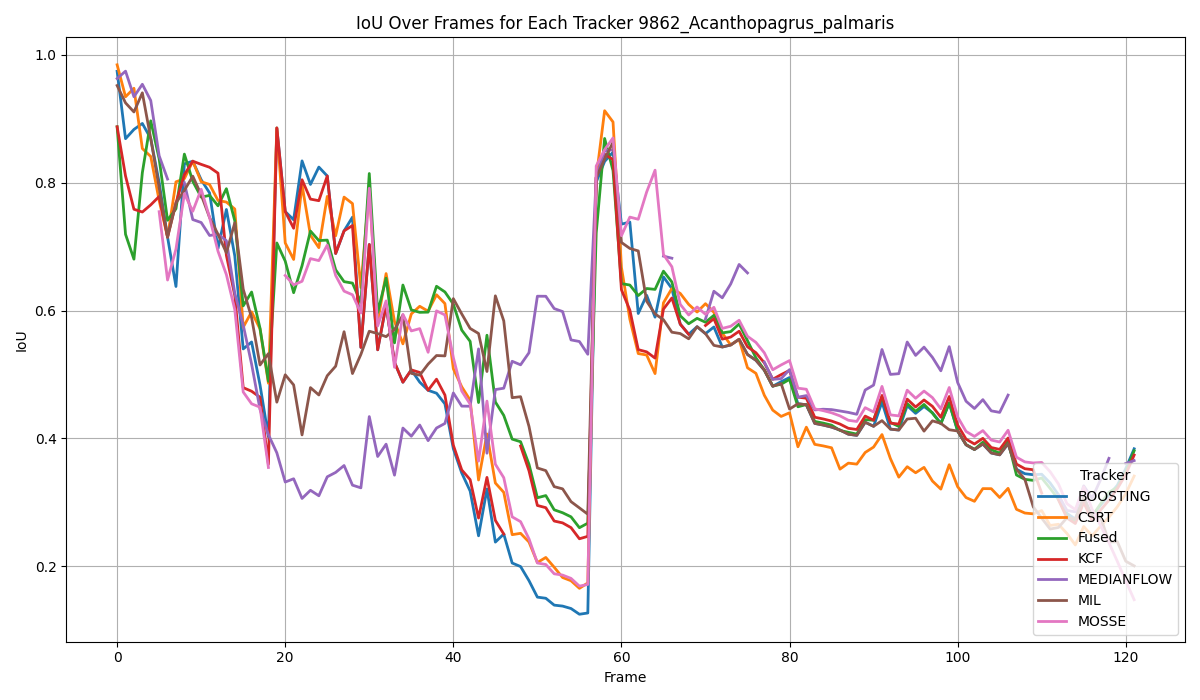


Figure A‑11: Video\_2 IoU over time for Tracker

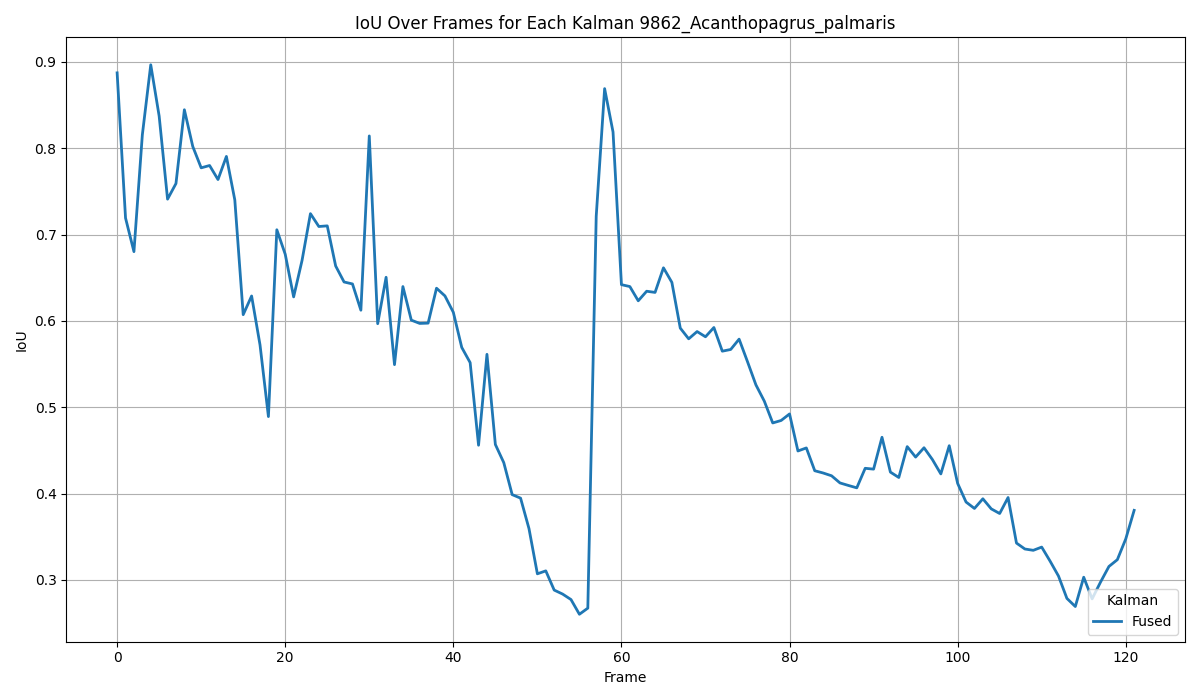


Figure A‑12: Video\_2 IoU over time for Kalman

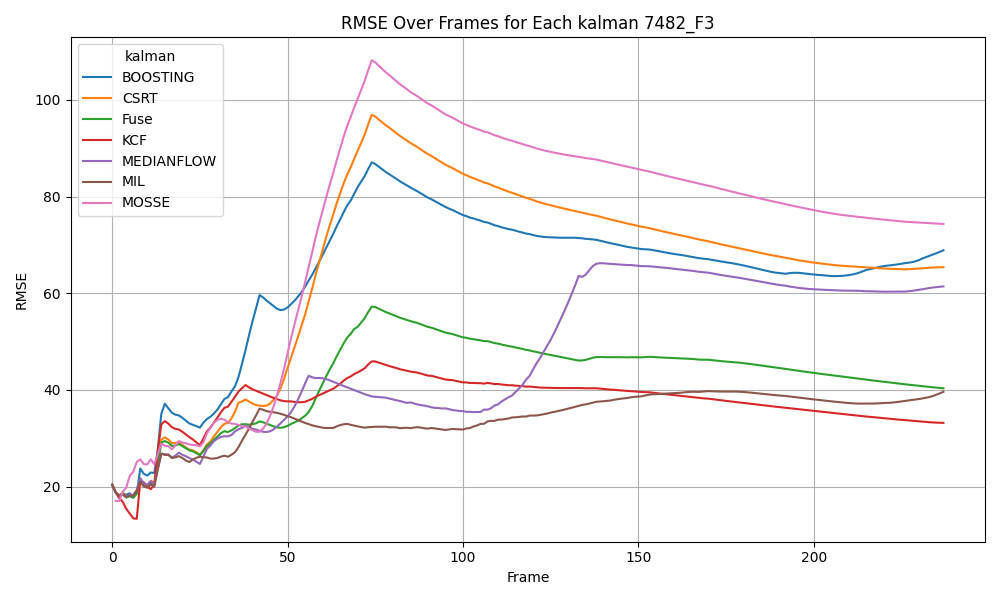


Figure A‑13: Video\_3 RMSE over time for Kalman

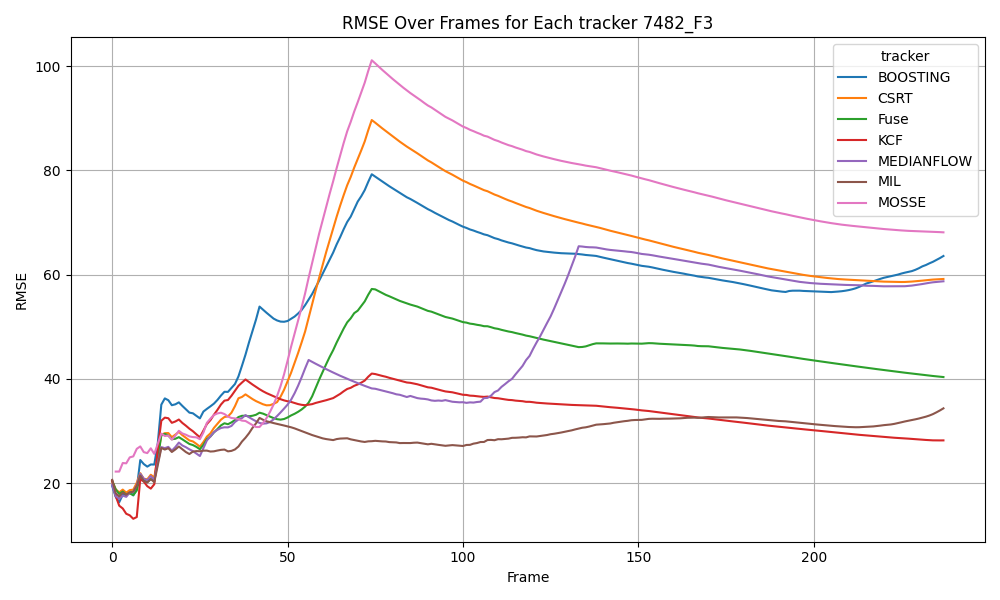


Figure A‑14: Video\_3 RMSE over time for Tracker

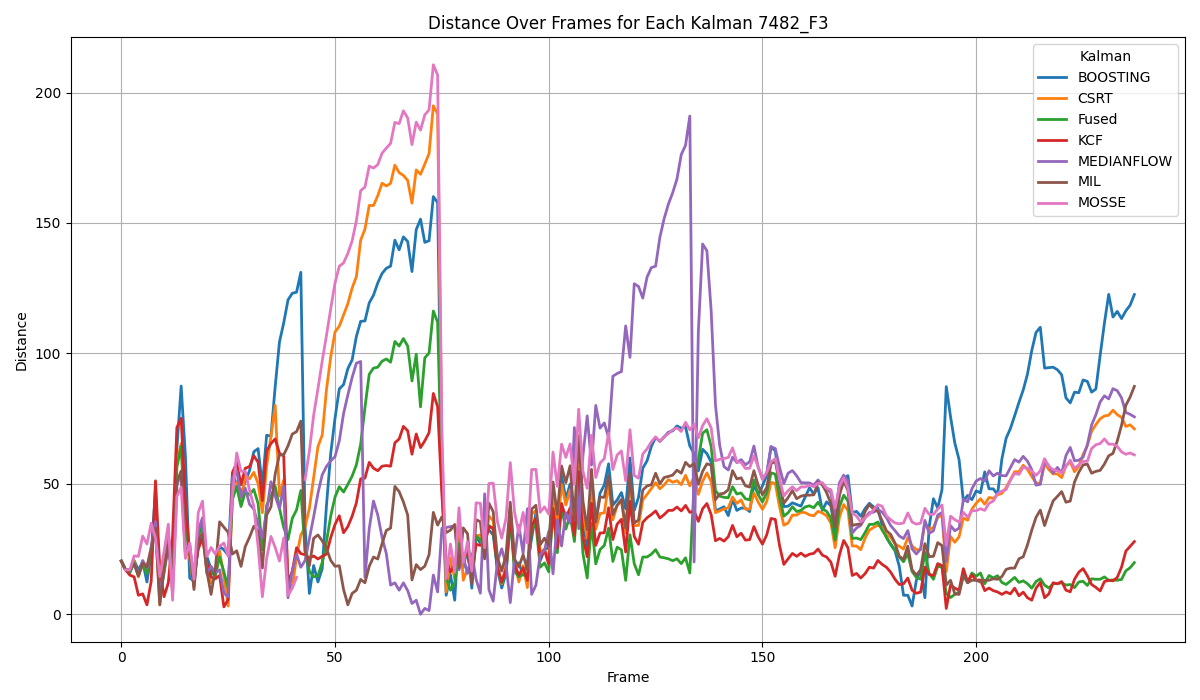


Figure A‑15: Video\_3 Euclidean distance over time for Kalman

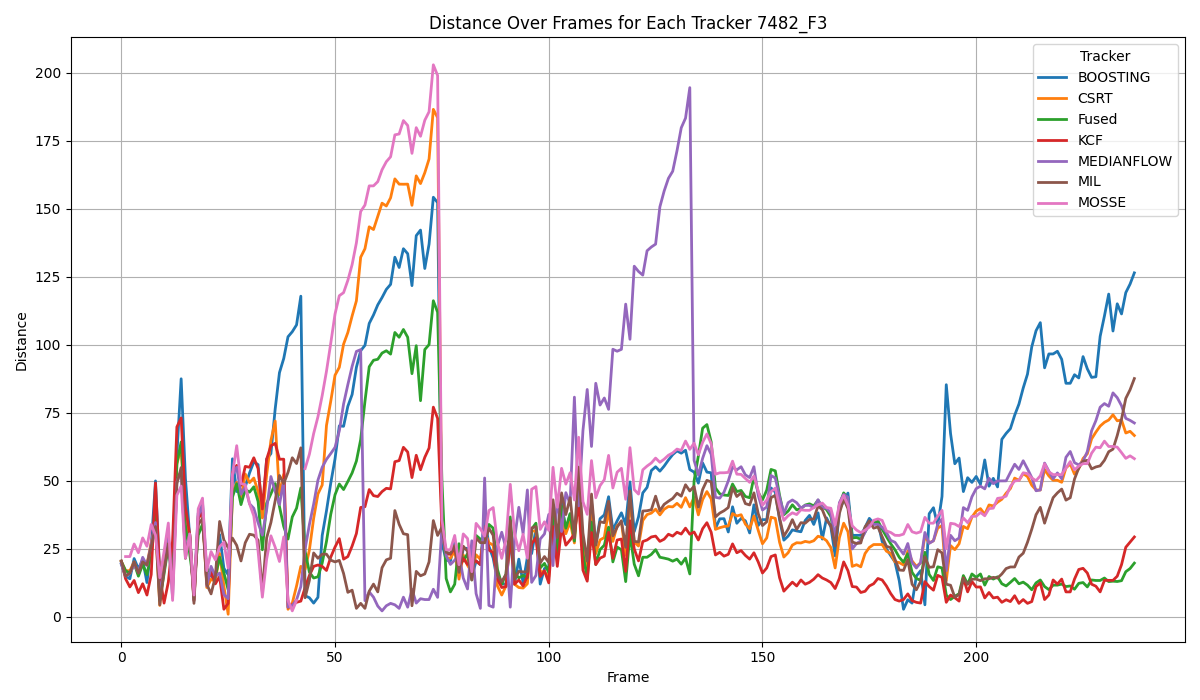


Figure A‑16: Video\_3 Euclidean distance over time for Tracker

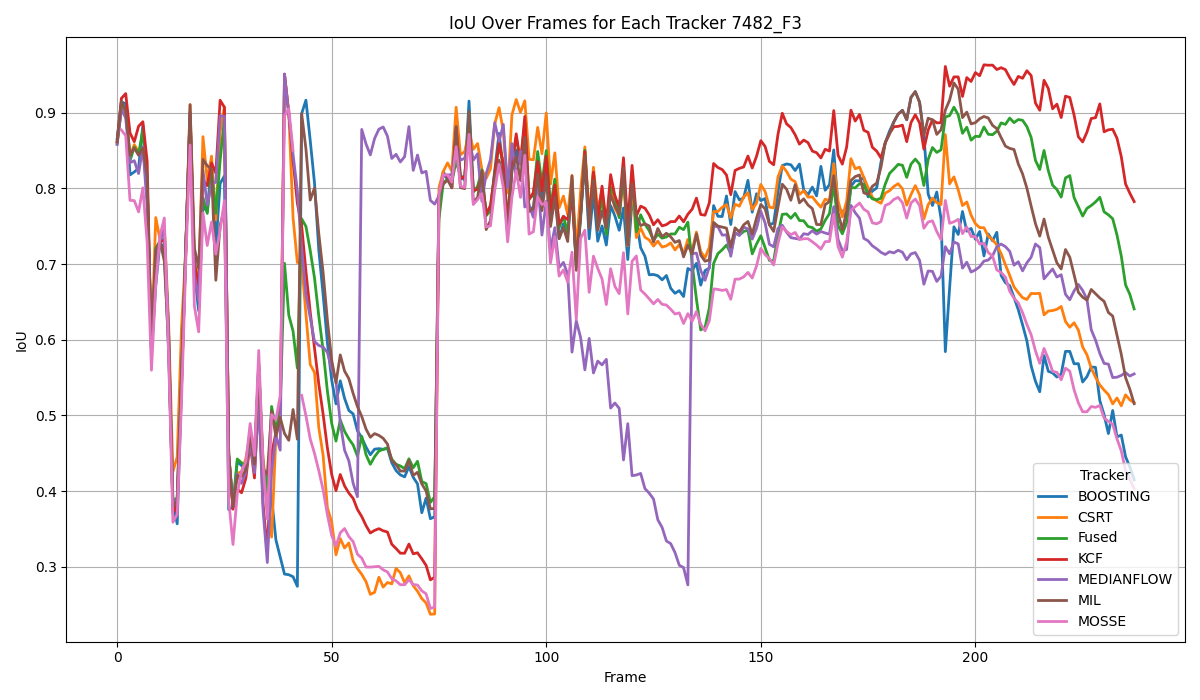


Figure A‑17: Video\_3 IoU over time for Tracker

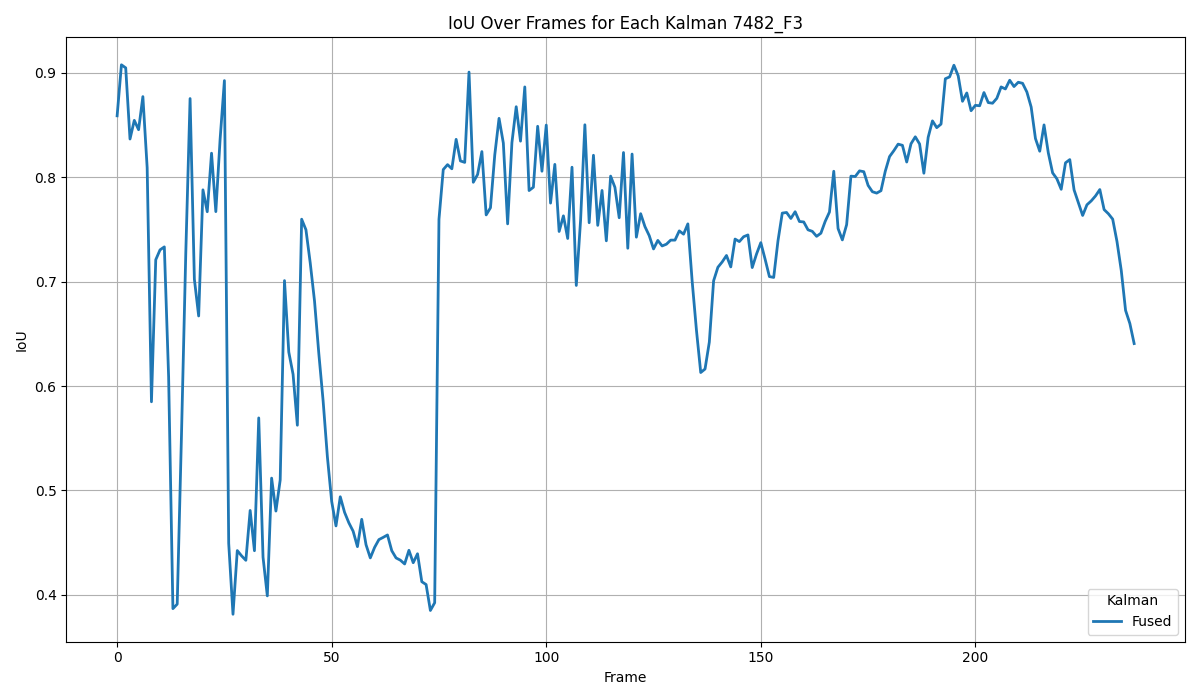


Figure A‑18: Video\_3 IoU over time for Kalman