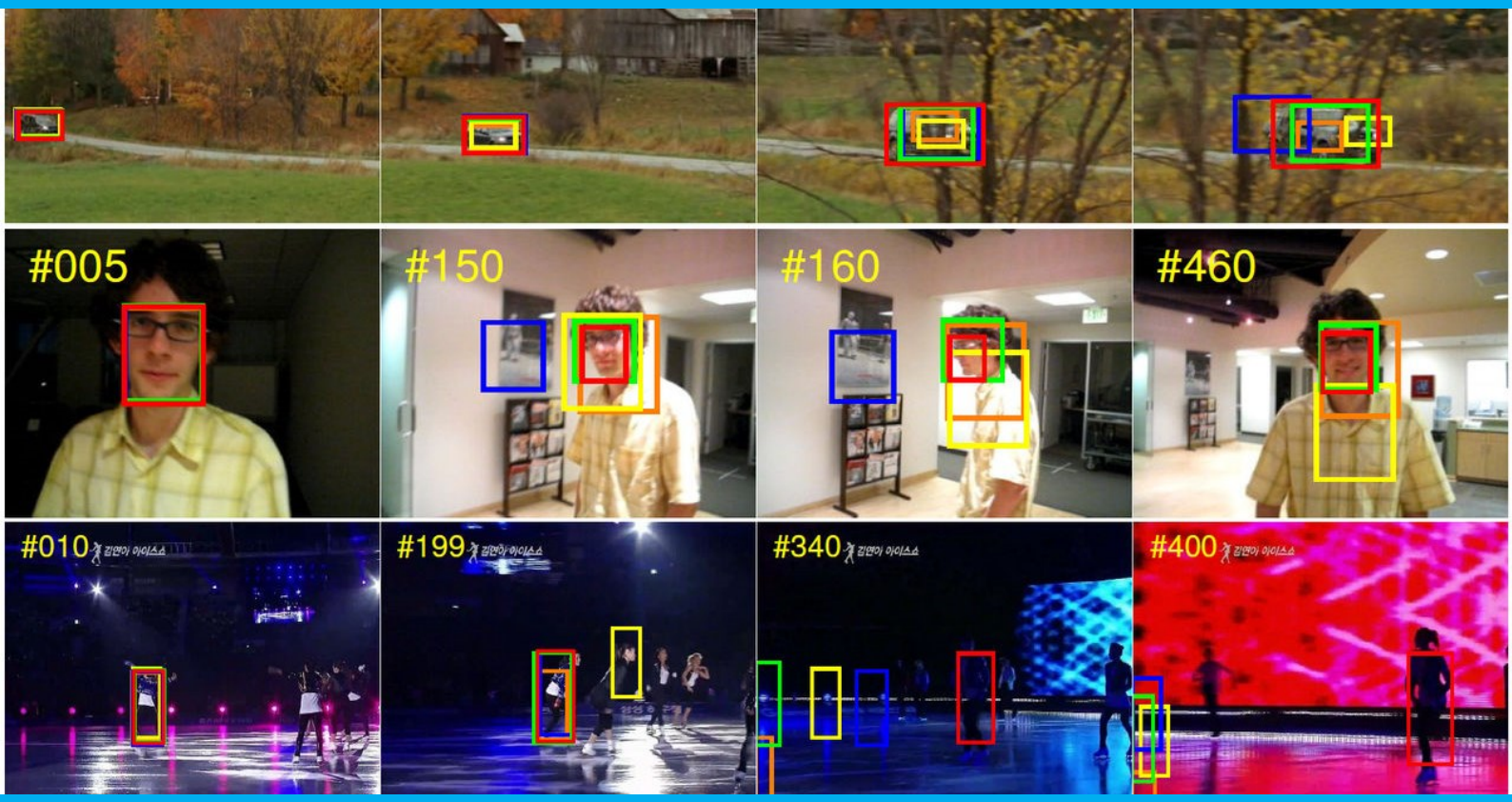
Master Thesis

Simple Online Visual Object Tracker Fusion based on Distributed Kalman Filtering

Author: Yigen Zhong



TU Delft

Master Thesis

Simple Online Visual Object Tracker Fusion based on Distributed Kalman Filtering

by

Author: Yigen Zhong

Student number: 5221919
Project duration: November 2, 2022 – April 10, 2023
Supervisors: Dr. J. Kooij, TU Delft, Primary supervisor
Dr. N. Tomen, TU Delft, Daily and Other supervisor

TU Delft

Abstract

Tracker-level fusion (TLF) is recognized as an effective approach to comprehensively improve visual object tracking performance by combining the capabilities of multiple baseline trackers. Although there is considerable interest in TLF, there are still challenges related to insufficient understanding, high cost, and unstable performance that make studying TLF difficult. In this thesis, I begin with an explicit summary of the overall pipeline of TLF which has significant guidance for TLF study. Additionally, I conduct a deep analysis of the positive and negative effects of baseline trackers to fully understand their influence on TLF. For visual object tracking, I propose three TLF frameworks based on three Distributed Kalman filters which are optimized for different scenarios and enable the fusion of different baseline trackers to enhance tracking performance. My TLF frameworks fuse the tracking results of different baseline trackers based on the principle of minimal trace to produce fusion results. Additionally, they exhibit superior and stable performance with general baseline tracker requirements, while also being simple, online, and real-time. Furthermore, the proposed frameworks can benefit from state-of-the-art baseline trackers over time, which will further improve their tracking performance. The proposed analysis and frameworks are studied extensively on three challenging benchmarks: generic tracking OTB2015, short-term tracking GOT-10k, and long-term tracking LaSOT. At over 240 FPS, the state-of-the-art success AUC score of 72.7% is achieved on OTB2015.

Keywords: Visual object tracking, Tracker-level fusion, Distributed Kalman filtering

Nomenclature

- VOT: Visual Object Tracking
- MOT: Multiple Object Tracking
- IoU: Intersection over Union
- KF: Kalman Filter
- TLF: Tracker-Level Fusion
- bbox: Bounding Box
- CF: Correlation Filtering
- FPS: Frames per Second
- CNN: Convolutional Neural Network
- FC: Fully Connected Layer
- ROI: Region of Interest
- DL: Deep Learning
- CV: Linear Constant Velocity
- CP: Linear Constant Position
- AUC: Area Under Curve
- GT: Ground Truth
- std: Standard Derivation
- RL: Reinforcement Learning

Contents

1 Introduction

1.1. Background

Visual Object Tracking (VOT), also known as Single Object Tracking, is an essential task in computer vision and has wide application in domains, including video surveillance (Brunetti et al., 2018; Tang et al., 2017; Haritaoglu et al., 2000), human-robotic interaction (Bonin-Font et al., 2008), and autonomous driving (Lee and Hwang, 2015). Therefore, studying VOT is paramount as it facilitates the application of computer vision technology in practice. However, for VOT, only the initial target state (e.g., position and scale) is available for the tracker, which limits the target information and poses a considerable challenge for VOT. Nonetheless, this unique nature of initialization offers VOT methods various possibilities in real-life scenarios. Compared with model-based tracking, such as tracking-by-detection, which can only track objects with the same classes as those in the training set, a significant advantage of VOT methods is that they can track objects of different classes, which can be classified as model-free tracking. This advantage of VOT methods provides an excellent solution to track some special targets, such as a person wearing clothing with a particular style in a crowd which is hard to obtain a sufficient dataset for training. However, there are many challenging factors for VOT tasks in real scenes, such as occlusion, scale variation, fast motion, background clutters, illumination variation, deformation, and more (Wu et al., 2015), as shown in Figs.1.1-1.5.

Figure 1.1: Occlusion: the target is partially or fully occluded. The top-left number is the frame step of the sequence.

1.2. Motivation

Nowadays, more and more scholars are interested in VOT with numerous significant works proposed. Generally, I divide VOT methods into two types based on the number of trackers in the tracking system: single tracker systems and multiple tracker systems. The former is more common for the VOT task. For example, now the most famous VOT method is the Siamese Network Tracking family, while previously popular trackers were based on Correlation Filtering (CF). These methods concentrate on enhancing the robustness of target appearance representation and mitigating inaccurate Bounding Box (bbox) prediction within a single tracker to address appearance and scale changes in a target. Undoubtedly, this is the basis for solving the VOT task, and it will still be the mainstream research direction in the

Figure 1.2: Scale Variation: the ratio of the bounding boxes of the first frame and the current frame is out of the range 2.

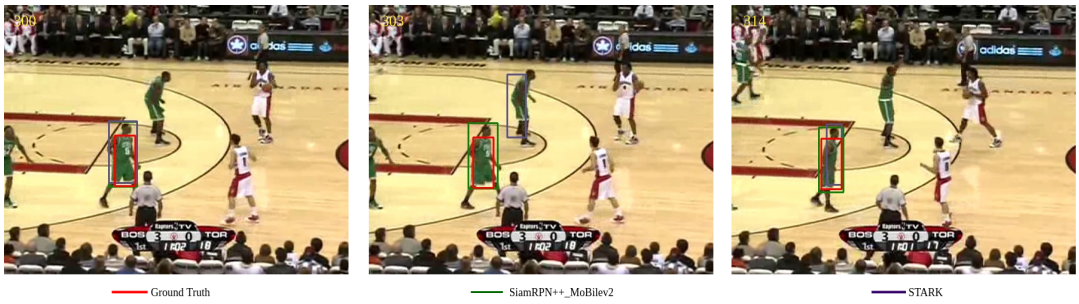


Figure 1.3: Background Clutters: the background near the target has a similar color or texture as the target.

Figure 1.4: Illumination Variation: the illumination in the target region is significantly changed.

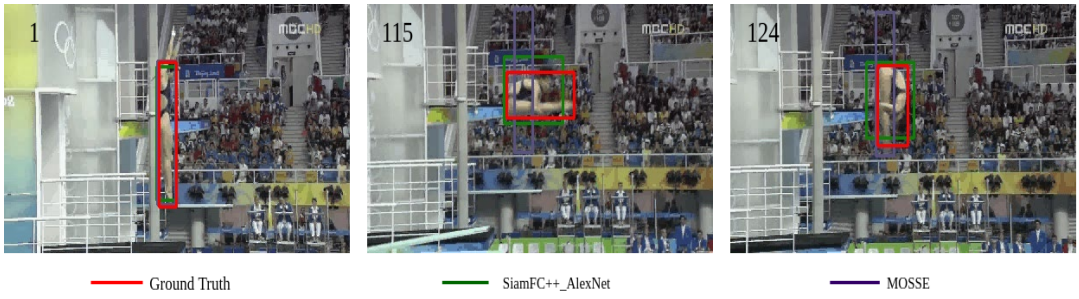


Figure 1.5: Deformation: non-rigid object deformation.

future. Nonetheless, as demonstrated in Figs.1.1-1.5, different trackers exhibit different tracking performances on various tracking scenes, making it challenging to identify a single tracker that performs well in all scenarios. To improve tracking performance, a simple, yet reasonable solution is to construct a tracking system that comprises multiple baseline trackers to leverage their strengths and circumvent their limitations. This methodology is called Tracker-level Fusion (TLF) or ensemble-based tracking (N. Wang et al., 2015). Here, TLF refers to the methods based on the tracking results of baseline trackers. Driven by this notion, several TLF methods have emerged to provide more reliable tracking performance by fusing the tracking results of different baseline trackers in recent times.

Despite the usefulness of TLF methods, questions remain concerning whether they adequately address the critical components of a TLF method, and whether the impact of tracking results from baseline trackers is adequately diagnosed. TLF methods tend to focus on proposing new approaches, leading to insufficient exploration of the above problems. It is worth noting that understanding the critical components has guiding significance on how to present a new TLF method. And diagnosing the influence of the baseline trackers is also crucial since the fusion result is based on the tracking results of the baseline trackers. Furthermore, while single-tracker-related work has seen vigorous development, the research on TLF is relatively limited. To enrich VOT task methods, it is essential to actively explore the TLF topic. While contemplating the proposal of a new TLF method, I discover various works (single tracker systems) that utilize Kalman Filtering (KF) (Kalman, 1960) for the VOT tasks, such as Rais and Munir (2021) and Iqbal et al. (2022). However, no work combines Distributed KF with TLF, where Distributed KF is a variant of KF that requires multiple trackers to collaboratively estimate target state. As KF plays an important role in practical engineering, investigating the relevant topic of combining Distributed KF with TLF idea for the VOT task is essential.

1.3. Objectives

Motivated by the aforementioned analysis, I propose the following research questions to be investigated in this thesis:

1. What are the essential components for designing and implementing a TLF method?

2. How do the baseline tracker's tracking results affect the accuracy and robustness of the TLF method's fusion result?

3. How can Distributed Kalman Filter be leveraged to propose a TLF method that improves tracking accuracy and robustness?

1.4. Contributions

In this thesis, I aim to answer the above research questions. To address the first research question, I deconstruct TLF methods into four constituent components: Tracker Ensemble, Tracker Evaluation, Result Fusion, and Feedback. Here, these four capitalized words above refer to the components. These components are commonly found in existing TLF methods. For the second research question, I conduct an in-depth analysis of baseline trackers to identify the positive and negative effects of their tracking results. The positive effect is tracker diversity, while negative effects include the limited number of baseline trackers, acceptability of the baseline tracker for feedback, occasional poor tracking results, and inevitable poor tracking results.

Concerning the third research question, I first propose the Standard Flexible Optimal KF (FOKF-Standard) for the VOT scenario based on our previous work (Zhong and Liu, 2021). Based on the above works for the first research question and Standard FOKF, I design a new TLF method, namely, TLF-FOKF-Standard. It considers the tracking results of baseline trackers as the local and community measurements before generating the fusion result. This fusion result is then fed back into baseline trackers to improve the stability of their tracking results. However, since FOKF-Standard only performs better for baseline trackers with similar tracking uncertainty which relies on the idea of average, I upgrade FOKF-Standard to FOKF-R by modifying the weight state vector. FOKF-R is effective in dealing with baseline trackers with significant differences in tracking uncertainty as long as there are accurate tracking uncertainties. Building upon the same strategy as TLF-FOKF-Standard, I present TLF-FOKF-R based on FOKF-R. To reduce the time complexity and exploit vague prior knowledge that only the

best baseline tracker in the tracker ensemble is known, I design FOKF-M by compressing the number of KF trackers of FOKF and propose TLF-FOKF-M similarly.

Overall, this thesis proposes a systematic approach for designing and implementing TLF methods by deconstructing them into essential components, analyzing the effects of baseline tracker results, and leveraging the capabilities of Distributed KF to improve tracking accuracy and robustness. My contributions can be summarized in four-fold:

1. By analyzing the existing TLF methods, I summarize four critical components of a TLF method with a general pipeline, which can guide the practical design of modern TLF methods and improve our understanding of TLF methods.

2. I identify the key characteristics of baseline trackers and their impact on the fusion result, offering important insights for the development and application of TLF methods in practical engineering.

3. I employ FOKF techniques on VOT and propose three simple but powerful TLF-FOKF methods that bridge the gap in Distributed KF applications for VOT.

4. My approach achieves state-of-the-art performance on three challenging benchmarks. To the best of my knowledge, my TLF-FOKF is the first TLF method that achieves success AUC score 72.7% on OTB2015 while running over 240 FPS.

In summary, Chapter 2 provides an overview of related works, including the general VOT tracker framework, baseline trackers, TLF methods, Distributed KF, and a summary. Chapter 3 presents the critical components of a TLF method. Additionally, this chapter discusses the influence of baseline trackers and proposes related TLF-FOKF methods. In Chapter 4, extensive experiments are performed to evaluate the effectiveness of the proposed analysis and methods thoroughly. The results demonstrate the superiority of the proposed TLF-FOKF methods. Finally, Chapter 5 concludes the thesis by summarizing the main findings and contributions, and offering recommendations for future research directions.

2

Related Work

The previous chapter has introduced the motivation and research questions of this thesis, but further details are required. In this chapter, I introduce a general VOT tracker framework that describes the common tracking mechanism used by baseline trackers and TLF methods. Then, I discuss common baseline trackers used in VOT and briefly describe their methodologies. The chapter also reviews state-of-the-art TLF methods and provides a description of Distributed KF to establish its feasibility in TLF. Finally, I summarize the key findings of this chapter, specifically highlighting important observations and research gaps addressed by my thesis.

2.1. A General VOT Tracker Framework

VOT is an important research topic in the field of computer vision, image understanding, and pattern recognition. The primary objective of VOT is to automatically obtain the states of the object in the subsequent video frames by giving the initial state (center location and scale) of an object in the first frame. N. Wang et al. (2015) proposed a general VOT tracker framework that summarizes the tracking mechanism of the existing tracking methods in the following steps: 1. initializing the observation model with a bbox of IT in the first frame; 2. generating candidate regions through the motion model; 3. extracting features of the candidate regions by the feature extractor; 4. calculating their probabilities using the observation model and selecting the region with the highest probability as an estimation of the IT object; 5. deciding whether the observation model requires an update based on its output; 6. using the ensemble post-processor to combine the bboxs of multiple trackers for a more accurate estimation. According to the these steps, the tracking methods based on the above five steps are defined as the baseline tracker, while the tracking methods involving step six are defined as the TLF methods.

2.2. Baseline Trackers for Visual Object Tracking

Based on the information provided in the previous section, it is evident that TLF methods rely on baseline trackers. Therefore, it is crucial to have an understanding of the methodology of baseline trackers to understand the role they play in TLF methods. The selection of baseline trackers is typically based on methodology categories and popularity.

2.2.1. Correlation Filtering-Based Tracking

Correlation filtering (CF) is one of the earliest VOT methods known for its simplicity and speed, achieved by applying a correlation filter to find the maximum response area of IT. MOSSE (Bolme et al., 2010) is a famous early work that achieved an ultra-fast tracking speed of 615 FPS using CF (Y. Zhang et al., 2021). Inspired by MOSSE, CSK (Henriques et al., 2012) proposed circulant matrices to achieve dense samples for improved performance at hundreds of FPS. However, neither of these methods could address the scale adaptive problem adequately, and their ability to express features was not strong enough. To overcome these limitations, SAMF (Y. Li and Zhu, 2014) integrated different features, and a scale pool was used to improve overall tracking capability. To further improve the ability to handle significant appearance variation in the object, LCT (Ma et al., 2015) introduced the temporal context

model to address large deformations and heavy occlusions. Boundary effects caused by sampling became more notable in the CF methods leading Dai et al. (2019) to propose a novel adaptive spatially-regularized correlation filters model to estimate better filter coefficients and spatial regularization weight. However, pursuing better performance has led to increased time complexity and the risk of overfitting in CF methods. ECO (Danelljan et al., 2017) revisited the core CF formulation, striking a balance between time complexity, accuracy, and robustness. Despite their initial popularity, recent years have seen minimal refreshing work in the field of CF.

2.2.2. Deep Learning-Based Tracking

Deep learning-based trackers are currently more popular due to the powerful feature representation capabilities of deep neural networks. Initially, scholars trained CNN for general feature representation and used fully connected layers (FC) to classify foreground and background or regress the location of IT, leading to models such as MDNet (Nam and Han, 2016), TCNN (Nam et al., 2016), and DSLT (Lu et al., 2018). However, these tracking models were bulky, with a low running speed. Similar to the way CF tracking worked and motivated by the strong feature representation abilities of CNN, Siamese Networks-based trackers summarized the VOT problem as measuring the similarity of the template of IT and ROI based on CNN feature representation. This approach significantly improved accuracy, running speed, and robustness. The pioneering work for VOT is SiamFC (Bertinetto et al., 2016). Several works have been proposed to improve the feature representation, scale adaptation, and temporal context, including GOTURN (Held et al., 2016), SiamRPN (B. Li et al., 2018), SiamRPN++ (B. Li et al., 2019), SiamFC++ (Xu et al., 2020), and STMTrack (Fu et al., 2021). However, CNN-based trackers have limited long-term tracking capabilities as their correlation-based feature-matching process cannot handle object appearance changes in long tracking sequences. As a result, there has been increasing interest in using transformers for object tracking, leveraging the development of the Attention mechanism (Vaswani et al., 2017). STARK (Yan et al., 2021) and SwinTrack (L. Lin et al., 2021), among others, have shown superior performance on long-term tracking by exploiting temporal context. However, the transformer's acquisition of local information is not as strong as CNN, and tracking continuity is challenging due to the encoding of location information.

2.3. Tracker-Level Fusion

As stated in the introduction of the previous section, the significant developments of baseline trackers have led to a considerable diversity of these trackers. Each has its own strengths and drawbacks in different VOT scenarios. TLF methods can absorb the strengths of different baseline trackers to further improve tracking performance. Therefore, this section discusses existing TLF methods to understand the current research status and identify research gaps.

Tracker-level fusion (TLF), also known as ensemble-based tracking, involves implementing multiple baseline trackers to achieve better tracking performance (Biresaw et al., 2014). A commonly employed TLF framework for enhancing feature representation and improving performance involves using multiple baseline trackers, each with distinct features (Yoon et al., 2012; N. Wang et al., 2018; Moorthy and Joo, 2021). Although effective, this technique's limitations arise from being exclusive to feature representation techniques.

Therefore, in an effort to extract more information from the baseline trackers and be more versatile, other researchers have proposed a bbox majority voting approach directly from the tracker level. One notable study, (Bailer et al., 2014), required only bbox inputs and introduced a loss function for bbox majority voting. This method essentially converts the fusion problem into maximizing energy or minimizing entropy by evaluating pairwise distances between different bbox of various baseline trackers, as demonstrated by J. Zhang et al. (2014) and Xie et al. (2019). This approach, however, often necessitates a large number of baseline trackers to achieve stable performance, leading to increased usage costs.

In contrast to the approach of using more baseline trackers to improve performance, some researchers are investigating how to efficiently fuse a small number of baseline trackers. For instance, Dunnhofer et al. (2022) presented an example of the complementary relationship between STARK and SuperDiMP (Bhat et al., 2019), where STARK produces bboxs that tightly fit the target, but with inconsistent tracking confidence (either wrong or overconfident). In contrast, SuperDiMP provides less accurate target localizations, but its tracking confidence is more consistent. Leveraging the comple-

mentary strengths of these two baseline trackers, the authors propose a tracker evaluation function to determine which tracker is correctly following the target. However, this approach is heavily dependent on the selection of the baseline trackers.

Recently, as reinforcement learning (RL) has gained momentum, some new studies have proposed a tracker selection strategy based on RL, including Dunnhofer et al. (2020) and W. Zhang et al. (2020). However, using a new baseline tracker requires additional training, which poses a significant challenge compared to majority voting. Additionally, RL-based algorithms are frequently unstable, especially when adapting to new environments.

2.4. Distributed Kalman Filtering

This section provides an overview of Distributed Kalman Filtering and explores its potential for use in TLF. Distributed KF is a variant of KF that differs in that it is designed for use in a distributed system where sensors and estimators operate independently and exchange information. Distributed KF is composed of multiple KF trackers that collaborate to estimate target states for improved estimation performance. Numerous works in the field of automatic control engineering have used Distributed KF for typical state estimation, including Zhong and Liu (2021), Yonggui and Bugong (2012), G. Wang et al. (2017), and Zhao and Guo (2017). One advantage of Distributed KF is its ability to optimally utilize subsystem information for fusion estimation in linear systems with Gaussian noise, while also retaining the simplicity, speed, and stability of KF. Based on this analysis, it is possible to treat the baseline trackers as sensors to generate measurements for corresponding KF trackers, which could act as a TLF method. To the best of our knowledge, no work has explored the application of Distributed KF to VOT, making further research in this area highly significant.

2.5. Summary

To summarize Sections 2.1 and 2.2, current research on baseline trackers concentrates on improving the feature extractor, motion model, observation model, and model updater to enhance tracking performance. While these improvements have resulted in unique advantages for baseline trackers, it remains challenging to find a tracker that performs well in all scenarios. Therefore, TLF represents an effective way to enhance tracking accuracy and robustness by combining the strengths of multiple baseline trackers from a broader perspective.

Regarding the development and use of baseline trackers ensembles, there are several categories of TLF methods, each with its own limitations. TLF methods based on feature fusion, special baseline tracker ensembles, and RL suffer from limited potential tracking performance or high usage costs since each requires a specific tracker ensemble which makes it difficult to benefit from new baseline trackers. Additionally, the stability of these methods' fusion performance is questionable due to the lack of solid mathematical derivation explaining their fusion process. TLF methods based on majority voting require a large number of baseline trackers to maintain stable fusion performance. Therefore, this thesis proposes a solution to address these challenges by applying the principles of Distributed KF. The proposed approach offers a simple and fast method for obtaining superior fusion performance while maintaining low usage costs and significant potential for enhancements. Furthermore, this work fills the gap in the field of VOT by introducing the application of Distributed KF.

3

Methodology

In this chapter, my proposed overview pipeline of a TLF method with four critical components is presented in Section 3.1. My deep analysis of the influence of tracking results of baseline trackers is given in Section 3.2. I propose FOKF-Standard which is revised for VOT in Section 3.3, while its proposed TLF method, namely, TLF-FOKF-Standard, is introduced in Section 3.4. In Section 3.5, I present an upgraded version of FOKF-Standard, FOKF-R, to overcome its limitations when tracking uncertainty of baseline trackers is significantly dissimilar, followed by TLF-FOKF-R in Section 3.6. Given that it is challenging to obtain accurate measurement uncertainty values for each frame and each baseline tracker in practice, I propose FOKF-M and TLF-FOKF-M in Section 3.7 and Section 3.8 by selecting the best baseline tracker as the main baseline tracker to take advantage of easy-to-learn prior knowledge. Moreover, FOKF-M demonstrates significantly reduced time complexity from quadratic-level to linear-level.

3.1. Critical Components of A Tracker-Level Fusion Method

In this section, I summarize the general pipeline and critical components of TLF methods. After reviewing various TLF techniques, I decompose the methodology into four constituent parts, outlined in Fig.3.1. Their functions are detailed below. Most TLF methods can be viewed as variations of this pipeline, including the pioneering work of Bailer et al. (2014), where these four components are partially incorporated but not explicitly defined or summarized. Additionally, Xie et al. (2019) proposed a comparable pipeline, including Tracker Ensemble, Result Fusion, and Tracker Evaluation but omitting Feedback. Dunnhofer et al. (2022) also designed a similar pipeline, but with a unique Result Fusion module that selects one of the tracking results of baseline trackers as the fusion output. Similarly, my pipeline framework can be used to understand the works of Moorthy and Joo (2021), N. Wang et al. (2018), Vojir et al. (2016), and other related studies.

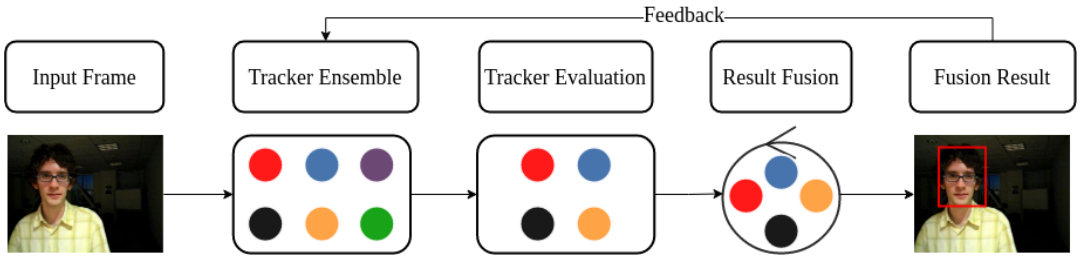


Figure 3.1: Overview pipeline of the proposed framework of most TLF methods. The various color dots correspond to different baseline trackers, while the surrounding rectangles depict the corresponding components involved.

1. **Tracker Ensemble**: This module comprises several baseline trackers that generate tracking results in parallel or sequentially from the same frame.

2. **Tracker Evaluation**: Its primary function is to assess the credibility of a tracker based on its

tracking results for Result Fusion. It determines whether the tracker's output passes to Result Fusion or not.

3. **Result Fusion**: This component weighs the tracking results and produces the final tracking output based on the credibility information derived from Tracker Evaluation.

4. **Feedback**: This module governs the strategy and frequency of updating the final result for the baseline trackers. It has to balance the trade-off between enhancing and hindering the tracking performance of the baseline trackers.

A tracking system that employs multiple baseline trackers typically initializes Tracker Ensemble, Tracker Evaluation, and Result Fusion with the given bounding box of the target in the initial frame. In each subsequent frame, the multiple baseline trackers in Tracker Ensemble generate candidate tracking results in parallel. Tracker Evaluation then evaluates these candidate results to compute their evaluation metrics, such as the probability of being the target. Based on the evaluation metrics, unreliable baseline trackers are removed from the ensemble, and the remaining tracking outputs are assigned their corresponding evaluation metrics. Result Fusion subsequently weights these tracking results based on the assigned evaluation metrics to produce the final result. If feedback is used to enhance the tracking performance of the baseline trackers and support the fusion process, the TLF algorithm is actively performed using the final result as its feedback. The pipeline is presented in Fig.3.1.

3.2. Deep Analysis of Baseline Trackers' Influence

Fig.1.1-1.5 clearly demonstrate the varying tracking results of different baseline trackers across different scenes. TLF methods attempt to achieve more accurate fusion results by skillfully leveraging the complementarity of each baseline tracker's tracking characteristics. However, inadequate knowledge of the baseline trackers' behavior may hinder the fusion process more than support it. This section presents how the tracking results of baseline trackers affect TLF in both positive and negative ways. The positive effect is the diversity that multiple trackers offer, while the negative effects include the limited number of baseline trackers, the baseline tracker's acceptability for Feedback, and occasional poor tracking results that are inevitable. Properly managing the positive effect can lead to a more stable and accurate fusion result, while neglecting the negative effects may hinder the overall performance improvement of TLF methods.

3.2.1. Positive Effect

Tracker diversity is a critical aspect to enhance the effectiveness of TLF methods N. Wang et al., 2015, which is defined as the overall performance of different baseline trackers. An essential evaluation metric that explicitly represents tracker diversity is the upper bound, as determined by selecting the tracking result of the baseline tracker with the highest GT-based IoU in each frame as the fusion output (Bailer et al., 2014). To demonstrate this idea clearly, we designed an ideal TLF algorithm that selects the best tracking result based on the GT-based IoU of baseline trackers in each frame, which generates the upper bound. An example of this outcome is shown in Fig.3.2. It can be observed that the tracker diversity of STMTrack and ECO_HC is significantly higher than that of STMTrack and SiamRPN, and their upper bounds manifest a positive correlation. This finding indicates that the improved tracking performance of TLF methods, as reflected by the higher Upper Bound curve, is mainly attributable to tracker diversity. Further and more comprehensive experiments are presented in Section 4.2.4.

3.2.2. Negative Effects

However, it is evident that introducing more baseline trackers enhances diversity but may also cause hindrances, as illustrated in Fig. 3.2; for instance, SiamRPN underperforms during frames [500,700], and STMTrack's performance deteriorates during frames [300,500]. Accordingly, it is imperative to identify the common negative impact of tracking results of baseline trackers for developing TLF approaches. To facilitate comprehensive comprehension of TLF methods, I advance four challenging factors, presented as follows. More comprehensive experimental results are shown in Sections 4.2.4 and 4.2.7.

1. **Limited Number of Baseline Trackers**: Tracker diversity is influenced by the number of baseline trackers used. For example, in Fig.3.2, STMTrack's average intersection-over-union (IoU)

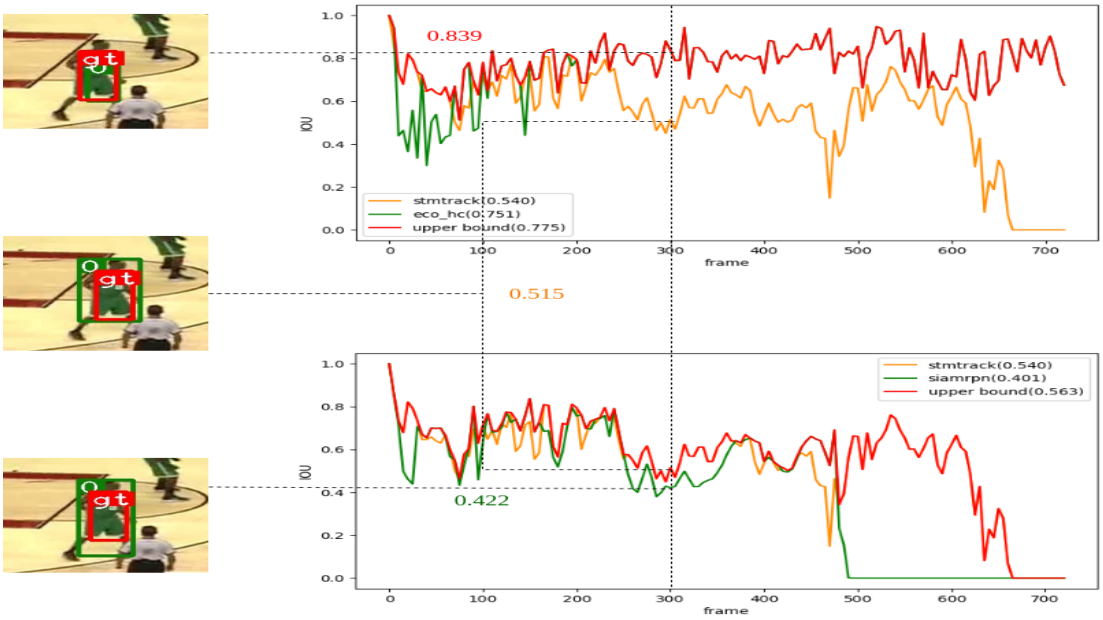


Figure 3.2: Tracker diversity. This figure illustrates the tracker diversity of STMTrack, ECO_HC, and SiamRPN along a sequence, as represented by the IoU curves. The corresponding simple TLF trackers are shown by the upper bound curves. The red rectangle represents the ground truth label, while the green rectangle indicates the tracking result. The red curve represents the upper bound curve, and the number after the legends is the average IoU. The top sub-figure (STMTrack+ECO_HC) exhibits a greater tracker diversity, resulting in a higher tracking performance for the corresponding TLF tracker (0.775). In contrast, the bottom sub-figure (STMTrack+SiamRPN) displays a lower tracker diversity, yielding a lower tracking performance for the corresponding TLF tracker (0.563).

is lower (0.515) than that of STMTrack+ECO_HC (0.839). Additionally, TLF methods require a minimum number of tracking results to achieve optimal performance. For instance, Xie et al. (2019) demonstrates that their TLF method performs better with three baseline trackers and has the best performance with eight, compared to using one or twenty. However, using too many baseline trackers presents challenges since they increase the computational, storage, and implementation costs, thus limiting the performance of TLF methods.

2. **Acceptability of Baseline Tracker for Feedback**: If a TLF algorithm requires feedback, the feasibility of the baseline tracker to support the fusion process must be taken into account. Bailer et al. (2014) reported that baseline trackers are usually not designed to receive feedback or be corrected by fusion results.

3. **Occasional Poor Tracking Results**: Occasional poor tracking results occur randomly with no discernable pattern. For instance, the reason for the poor performance of STMTrack and SiamRPN during frame 300 is unknown, although they perform reasonably well at some other times.

4. **Inevitable Poor Tracking Results**: Inevitable poor tracking results occur when a baseline tracker is limited in some scenes. For example, as shown in Fig.1.5, MOSSE performs poorly in scenes with drastic scale changes since it does not have scale changeability. DeepSORT is limited by a fixed set of classes and object bbox sizes based on the training set, which explains its poor performance on unrestricted test sets. Fig.1.4 shows that DeepSORT tracks the whole singer instead of just the upper body for VOT tasks, caused by misalignment of training and test sets. STARK jumps repeatedly in a large range when encountering similar targets (Fig.1.3), making it difficult to track objects effectively sometimes.

3.3. Standard Flexible Optimal Kalman Filtering

In this section, I describe my proposed Standard Flexible Optimal KF (FOKF-Standard) in detail. The idea is derived from our previous work (FOKF) (Zhong and Liu, 2021), and the equations of FOKF-

Standard are similar to FOKF. However, due to the different measurement definitions of Z_t^i, there are slight differences in their corresponding equations. In FOKF, Z_t^i is defined as $Z_t^i = a_t^i H_t^i X_t + v_t^i$ since it incorporates the actual sensor noise of the sensor. On the other hand, FOKF-Standard defines Z_t^i as Eq.3.2 as it does not need to account for the noise of the baseline tracker for VOT tasks if the tracking result is invalid.

FOKF-Standard is an extension of FOKF that incorporates the refinement proposed in Eq.3.2. It is a distributed Kalman filter that can optimally estimate the state of the target across intermittent observations and varying sensing states. Based on the minimum error covariance trace principle, FOKF-Standard can optimally and collaboratively estimate the state of the target via their own and community observations, which derive the local Kalman gain and the community Kalman gain, respectively. In addition, the design of the flexible values can satisfactorily represent the various circumstances, including partial trackers lost, all trackers lost, and no trackers lost. Here, the tracker lost denotes that the baseline tracker provides an invalid measurement. With a limited number of trackers in the tracker ensemble, which is usually constrained by hardware and labor costs, FOKF-Standard has a low time complexity of $O(NM)$ with N and M representing the number of baseline trackers and the community of one of the baseline trackers, respectively.

3.3.1. Problem Formulation

In a TLF method, the initial step involves the use of a baseline tracker ensemble to generate tracking results (measurements). Based on the nature of the VOT task, the following notation and assumptions can be made:

Notation: For a baseline tracker ensemble that provides measurements, a corresponding KF tracker ensemble can be created. Let $T_t = \{1, 2, \ldots, N\}$ denote the KF tracker ensemble at time step t, whereby $i \in T_t$ represents the index of KF tracker s_i. The KF tracker s_i is connected to a set of indices representing its community, denoted $C_t^i = \{j \in T_t : j \neq i\}$. In VOT tasks, there is no limitation on the interchange of information among KF trackers, thus allowing s_i to freely exchange information with other KF trackers.

Assumption 1:a_t^i, c_t^i, and d_t^i are defined as the flexible values. Here, $a_t^i = 1$ if the measurement of s_i is valid, otherwise, $a_t^i = 0$. $c_t^i = 1$ if its community C_t^i has the element j's $a_t^j = 1$ or s_i's $a_t^i = 1$, otherwise, $c_t^i = 0$. $d_t^i = 1$ if $a_t^i = 0$ and $c_t^i = 0$, otherwise, $d_t^i = 0$. $n_t^i = \sum_{j \in C_t^i} a_t^j$. Note that a_t^i is used to indicate whether measurement Z_t^i is valid, c_t^i is used to represent whether there is more than one valid measurement, d_t^i is used to guarantee FOKF-Standard to be normal in extreme cases (without valid measurements), n_t^i is used to count the number of valid community measurements.

The movement of the target in VOT tasks can be represented as a linear discrete time-varying system with Gaussian Distributed noise w_t as follows:

$$X_t = A_t X_{t-1} + w_t \tag{3.1}$$

Because the sensing noise does not need to be considered in the VOT task if the measurement is invalid, the measurement Z_t^i can be defined as follows:

$$Z_t^i = a_t^i (H_t^i X_t + v_t^i) \tag{3.2}$$

where $X_t \in R^n$ is the true state vector of IT, $Z_t^i \in R^m$ is the measurement vector received by s_i. $A_t \in R^{n \times n}$ is the state transition matrix, while $H_t^i \in R^{m \times n}$ is the state-to-measurement matrix. $w_t \in R^n$ is the processing Gaussian Noise with zero mean and covariance matrix Q_t, while $v_t \in R^m$ is the measurement Gaussian Noise with zero mean and covariance matrix R_t^i.

The designed filter is shown as follows:

$$e_t^i = a_t^i(Z_t^i - H_t^i X_{t|t-1}^w) \tag{3.3}$$

$$X_{t|t}^i = (c_t^i + d_t^i)X_{t|t-1}^w + a_t^i L_t^i e_t^i + N_t^i \sum_{j \in C_t^i} e_t^j \tag{3.4}$$

$$X_{t+1|t}^i = A_t X_{t|t}^i \tag{3.5}$$

$$X_{t|t}^w = \frac{\sum_{i \in T_t} X_{t|t}^i}{\sum_{i \in T_t} c_t^i + d_t^i} \tag{3.6}$$

$$X_{t+1|t}^w = \frac{\sum_{i \in T_t} X_{t+1|t}^i}{\sum_{i \in T_t} c_t^i + d_t^i} \tag{3.7}$$

where $e_t^i \in R^m$ is the residual error, $X_{t|t}^i \in R^n$, $X_{t+1|t} \in R^n$ and $X_{t+1|t}^w$ are the update, prediction, weight update, and weight prediction estimation of s_i, respectively. L_t^i and $N_t^i \in R^{n \times m}$ are called local Kalman gain and community Kalman gain, respectively.

3.3.2. Derivation

According to Eqs.3.1, 3.4, 3.5 and 3.7, the corresponding estimation errors are defined as:

$$\widetilde{X}_{t+1|t}^i \equiv (c_t^i + d_t^i)X_{t+1} - X_{t+1|t}^i \tag{3.8}$$

$$\widetilde{X}_{t|t}^i \equiv (c_t^i + d_t^i)X_t - X_{t|t}^i \tag{3.9}$$

$$\widetilde{X}_t \equiv X_t - X_{t|t-1}^w \tag{3.10}$$

Based on Eqs.3.1, 3.5, 3.8, it can derive as follows:

$$\widetilde{X}_{t+1|t}^i = A_t \widetilde{X}_{t|t}^i + (c_t^i + d_t^i)w_t \tag{3.11}$$

For Eqs.3.2-3.4 and 3.9-3.10, the update estimation error is derived as follows:

$$\widetilde{X}_{t|t}^i = [(c_t^i + d_t^i)I_p - a_t^i L_t^i H_t^i - n_t^i N_t^i H_t^i]\widetilde{X}_t - a_t^i L_t^i v_t^i - N_t^i \sum_{j \in C_t^i} a_t^j v_t^j \tag{3.12}$$

where $n_t^i = \sum_{j \in C_t^i} a_t^j$ and I assume $H_t^i = H_t^j$.

The optimal estimation can be obtained by minimizing the following cost function:

$$J = \sum_{i \in T_t} E[\widetilde{X}_{t|t}^i (\widetilde{X}_{t|t}^i)^T] \tag{3.13}$$

According to Eqs.3.8 and 3.13, the prediction error edge-covariance is defined and derived as follows:

$$P_{t+1|t}^{i,j} \equiv E[\widetilde{X}_{t+1|t}^i (\widetilde{X}_{t+1|t}^j)^T] = A_t P_{t|t}^{i,j} (A_t)^T + (c_t^i + d_t^i)(c_t^j + d_t^j)Q_t \tag{3.14}$$

Because my goal is to minimize the cost function Eq.3.13, I assume $i = j$, and then the prediction error variance and the weight prediction error variance can be derived as follows:

$$P_{t+1|t}^{i,i} \equiv P_{t+1|t}^i = A_t P_{t|t}^i (A_t)^T + (c_t^i + d_t^i)Q_t \tag{3.15}$$

$$P_{t+1|t}^w \equiv E[\widetilde{X}_{t+1} (\widetilde{X}_{t+1})^T] = \frac{\sum_{i \in T_t} P_{t+1|t}^i}{\sum_{i \in T_t} (c_t^i + d_t^i)} \tag{3.16}$$

Similarly, the update error covariance can be obtained according to Eq.3.12:

$$\begin{aligned} P_{t|t}^{i,j} &\equiv E[\tilde{X}_{t|t}^{i}(\tilde{X}_{t|t}^{j})^{T}] \\ &= [F_t^i P_{t|t-1}^{w}(F_t^j)^T] + a_t^i a_t^j L_t^i R_t^{i,j}(L_t^j)^T + a_t^i L_t^i \sum_{s\in C_t^j} a_t^s R_t^{i,s}(N_t^j)^T \\ &+ N_t^i \left(\sum_{r\in C_t^i} a_t^r v_t^r \sum_{s\in C_t^j} a_t^s (v_t^s)^T \right)(N_t^j)^T + a_t^j N_t^i \sum_{r\in C_t^i} a_t^r R_t^{r,j}(L_t^j)^T \\ &= [F_t^i P_{t|t-1}^{w}(F_t^j)^T] + a_t^i a_t^j L_t^i R_t^{i,j}(L_t^j)^T + N_t^i \sum_{r\in C_t^i \cap C_t^j} a_t^r R_t^{r,r}(N_t^j)^T \end{aligned} \tag{3.17}$$

where $F_t^i = (c_t^i + d_t^i)I_p - a_t^i L_t^i H_t^i - n_t^i N_t^i H_t^i$. Because the measurement noises are mutually independent, it can get that $\sum_{s\in C_t^j} R_t^{i,s} = 0$, $\sum_{r\in C_t^i} R_t^{r,j}$ and $\sum_{r\in C_t^i} v_t^r \sum_{s\in C_t^j}(v_t^s)^T = \sum_{r\in C_t^i \cap C_t^j} R_t^{r,r}$.

Also, I assume $i = j$, Eq.3.17 is rewritten as below:

$$P_{t|t}^{i,i} \equiv P_{t|t}^{i} = [F_t^i P_{t|t-1}^{w}(F_t^i)^T] + a_t^i L_t^i R_t^i (L_t^i)^T + N_t^i \sum_{j\in C_t^i} a_t^j R_t^j (N_t^i)^T \tag{3.18}$$

Definition 1: A class of distributed KF with the flexible values, the local Kalman gain, and the community Kalman gain is defined as the Standard Flexible Optimal Kalman Filter (FOKF-Standard), which minimizes the trace $\sum_{i\in T_t} tr\{P_{t|t}^i\}$ and consists of the following Eqs.3.3-3.7, 3.15, 3.16, 3.18-3.21.

Theorem 1: According to Eqs.3.1, 3.2, 3.18 and Assumption 1, there exists the following local and community Kalman gains:

$$L_t^i = [a_t^i(c_t^i + d_t^i)P_{t|t-1}^w(H_t^i)^T(I_m - (n_t^i)^2 G_t^i S_t^i)\bar{S}_t^i][I_m - a_t^i(n_t^i)^2 S_t^i G_t^i S_t^i \bar{S}_t^i]^{-1} \tag{3.19}$$

$$N_t^i = [n_t^i(c_t^i + d_t^i)P_{t|t-1}^w(H_t^i)^T(I_m - a_t^i \bar{S}_t^i S_t^i)G_t^i][I_m - a_t^i(n_t^i)^2 S_t^i \bar{S}_t^i S_t^i G_t^i]^{-1} \tag{3.20}$$

where

$$\begin{cases} S_t^i = H_t^i P_{t|t-1}^w (H_t^i)^T \\ \bar{S}_t^i = [a_t^i(S_t^i + R_t^i)]^{-1} \\ G_t^i = [(n_t^i)^2 S_t^i + \sum_{j\in C_t^i} a_t^j R_t^j]^{-1} \end{cases} \tag{3.21}$$

Proof 1: Based on Assumption 1, the following properties can be obtained have been used in the above-related equations:

$$\begin{cases} a_t^i = (a_t^i)^2 \\ c_t^i + d_t^i = (c_t^i + d_t^i)^2 \\ E[a_t^i v_t^i (a_t^i v_t^i)^T] \equiv a_t^i R_t^i \end{cases} \tag{3.22}$$

The optimal local Kalman gain L_t^i and the community Kalman gain N_t^i can be solved by the following equations:

$$\begin{cases} \dfrac{\partial tr(P_{t|t}^i)}{\partial L_t^i} = 0 \\ \dfrac{\partial tr(P_{t|t}^i)}{\partial N_t^i} = 0 \end{cases} \tag{3.23}$$

Based on Eqs.3.18, 3.22, 3.23, there are:

$$\begin{aligned} \frac{\partial tr(P_{t|t}^i)}{\partial L_t^i} &= -2a_t^i(c_t^i + d_t^i)P_{t|t-1}^w(H_t^i)^T + 2a_t^i L_t^i H_t^i P_{t|t-1}^w(H_t^i)^T \\ &+ 2a_t^i n_t^i N_t^i H_t^i P_{t|t-1}^w(H_t^i)^T + 2a_t^i L_t^i R_t^i = 0 \end{aligned} \tag{3.24}$$

The optimal local Kalman gain L_t^i can be yielded by the above equation as:

$$L_t^i = a_t^i[(c_t^i + d_t^i)I_p - n_t^i N_t^i H_t^i]P_{t|t-1}^w(H_t^i)^T[a_t^i(H_t^i P_{t|t-1}^w(H_t^i)^T + R_t^i)]^{-1} \tag{3.25}$$

Similarly, solving Eq.3.26 yields the optimal community Kalman gain N_t^i as Eq.3.27:

$$\begin{aligned}\frac{\partial tr(P_{t|t}^i)}{\partial N_t^i} &= -2n_t^i(c_t^i + d_t^i)P_{t|t-1}^w(H_t^i)^T + 2a_t^i n_t^i L_t^i H_t^i P_{t|t-1}^w(H_t^i)^T \\ &\quad + 2(n_t^i)^2 N_t^i H_t^i P_{t|t-1}^w(H_t^i)^T + 2N_t^i \sum_{j\in C_t^i} a_t^j R_t^j = 0\end{aligned} \tag{3.26}$$

$$N_t^i = n_t^i[(c_t^i + d_t^i)I_p - a_t^i L_t^i H_t^i]P_{t|t-1}^w(H_t^i)^T[(n_t^i)^2 H_t^i P_{t|t-1}^w(H_t^i)^T + \sum_{j\in C_t^i} a_t^j R_t^j]^{-1} \tag{3.27}$$

Two Eqs.3.25 and 3.27 contain two unknown variables L_t^i and N_t^i, and thus, two equations with two unknown variables can be solved. The results are shown in Eqs.3.19-3.21.

3.3.3. Analysis

FOKF-Standard is a versatile algorithm that adapts to multiple patterns with varying degrees of flexibility. For example, FOKF-Standard is similar to the distributed optimal KF (Yonggui and Bugong, 2012) when $a_t^i = 1$, $n_t^i \neq 0$. FOKF-Standard is reduced to the single KF when $a_t^i = 1$, $n_t^i = 0$. FOKF-Standard is reduced to the open-loop KF when $d_t^i = 1$ which skips the update step of FOKF-Standard. In addition, considering the abnormal cases of $a_t^i = 0$ and $n_t^i = 0$, it can use a small value for Eq. 3.21 for calculation. The algorithm of FOKF-Standard is designed as Algorithm 1.

FOKF-Standard has several prerequisites that need to be met to achieve optimal performance. It requires the motion model of the target to be linear and the system noise to follow Gaussian Distribution. In addition, the key hyperparameters, namely A_t, Q_t, and R_t^i, must accurately represent the true states. Due to the average design of the weight estimation, FOKF-Standard performs best when $R_t^i \equiv R_t$. However, it can be challenging to fulfill all these prerequisites in real-life applications. Nevertheless, as long as the actual situation closely aligns with these prerequisites, FOKF-Standard can still provide reliable results. To evaluate the effectiveness of FOKF-Standard, we conducted extensive experiments in Sections 4.1.2 to 4.1.5.

Algorithm 1: FOKF-Standard for object tracking.

Initialize: $X_{0|-1}^w = X_0$, $P_{0|-1}^w = P_0$, set $T_t = \{\emptyset\}$ and $C_t^i = \{\emptyset\}$ if they will change, $a_t^i = c_t^i = d_t^i = n_t^i = 0$.
Input: Measurement vectors Z_t^i.
1: Update T_t and C_t^i if they change and update the flexible values (a_t^i, c_t^i, d_t^i) and n_t^i based on measurements.
2: Do the update step of FOKF-Standard by using Eqs.3.3, 3.4, 3.18-3.21.
3: Do the prediction step of FOKF-Standard by using Eqs.3.5, 3.15.
4: Do the weight step of FOKF-Standard by using Eqs.3.7, 3.16.
5: Update time step t, jump to *Step* 1, return $X_{t+1|t}^w$and $P_{t+1|t}^w$.

3.4. Tracker-Level Fusion Based on FOKF-Standard

The primary objective of this section is to propose TLF-FOKF-Standard for the VOT task. TLF-FOKF-Standard utilizes FOKF-Standard as the component Tracker Evaluation and Result Fusion of the TLF algorithm. The section provides an overview of the TLF-FOKF-Standard framework and defines the state space of the critical parameters of FOKF-Standard in detail.

3.4.1. Proposed Framework

Sections 3.1 and 3.3 establish that FOKF-Standard satisfies the requirements of the component Tracker Evaluation and Result Fusion of a TLF algorithm. Moreover, the component Tracker Ensemble is es-

sential for a TLF algorithm. As FOKF-Standard does not design the strategy of baseline tracker removal, the component Feedback is necessary to prevent the baseline trackers from generating too many outlier tracking results. Therefore, I propose TLF-FOKF-Standard by combining the four capitalized components mentioned above. This combination results in an optimal TLF algorithm as long as the prerequisites of FOKF-Standard are in place. The proposed framework is shown in Fig.3.3. The detailed design of each component is described below.

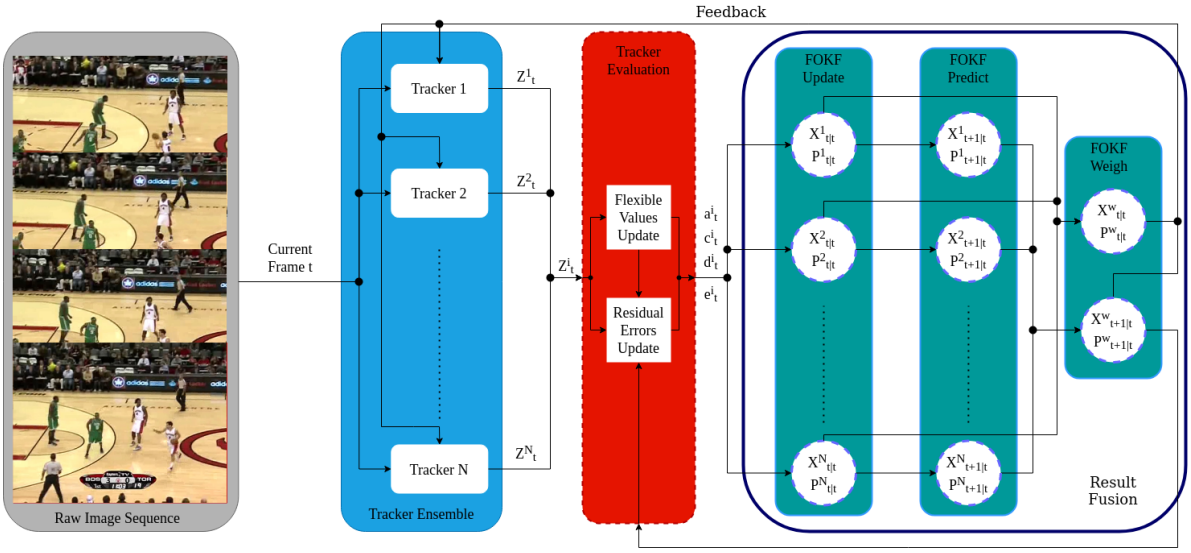


Figure 3.3: The primary framework of TLF-FOKF-Standard: the left side is Tracker Ensemble for generating measurement Z_t^i. The Tracker Evaluation component for updating flexible values and calculating residual errors lies in the middle. The Result Fusion based on FOKF-Standard is on the right side for updating IT's state vector X_t^i and state covariance matrix P_t^i. After acquiring the weight estimation states, they are fed back to Tracker Ensemble and the residual error update module for iteration. It is a typical TLF algorithm that contains the component Tracker Ensemble, Tracker Evaluation, Result Fusion, and Feedback. Feedback here is designed to help converge the tracking results of baseline trackers and alleviate the problem of outlier measurements.

Tracker Ensemble: To make the insertion of baseline trackers more straightforward and improve practicality, TLF-FOKF-Standard requires the baseline trackers to meet the following conditions:

1. The baseline tracker must be capable of performing the VOT task independently, including taking raw input images and outputting the IT states.
2. The baseline tracker must be able to update the intermediate states of the IT according to the feedback of FOKF-Standard, such as the weighted state. This requirement can be disregarded when feedback is unnecessary.

This design has the advantage of being compatible with most current baseline trackers, as the requirements are straightforward and practical. Users require only knowledge of the baseline tracker's interface to insert it into the Tracker Ensemble. Moreover, if a pre-trained tracker exists, users can use it directly without incurring the cost of retraining it. For the experimental evaluation in this study, we have selected nine baseline trackers that satisfy the requirements, ensuring the results are universal. The selected trackers are MOSSE, ECO, DeepSORT, GOTURN, SiamRPN, SiamRPN++, SiamFC++, STMTrack, and STARK, as presented in Table 3.1.

Tracker Evalution: The measurement vector Z_t^i is the input of this module and is modeled as Eq. 3.28, where $[x_c, y_c]$ are designed as the horizontal and vertical pixel location of the center of the bbox, while the scale w and h represent the width and height, respectively. After that, a_t^i is updated based on whether the Z_t^i is valid. And then, the flexible values c_t^i, d_t^i, n_t^i, and the residual error e_t^i are updated and output as the input of the Result Fusion. They contain information on trackers' states and credibility. H_t^i depends on the relationship between X_t^i and Z_t^i.

$$Z_t^i = [x_c, y_c, w, h]^T \tag{3.28}$$

Table 3.1: A overview of baseline trackers.

Baseline Tracker	Main Technology	Citation
MOSSE	CF	Bolme et al., 2010
GOTURN	DL(Siamese)	Held et al., 2016
ECO	CF	Danelljan et al., 2017
DeepSORT	DL(Tracking by Detection)	Wojke et al., 2017
SiamRPN	DL(Siamese)	B. Li et al., 2018
SiamRPN++	DL(Siamese)	B. Li et al., 2019
SiamFC++	DL(Siamese)	Xu et al., 2020
STMTrack	DL(Siamese)	Fu et al., 2021
STARK	DL(Self-Attention)	Yan et al., 2021

Result Fusion: Baseline trackers, denoted as s_i, are each associated with a KF tracker that has an estimation state vector X_t^i and an estimation state covariance P_t^i. The vector X_t^i is defined by the equation shown in Eq. 3.29. To perform the update step, local measurements and community measurements are employed to optimally solve for $X_{t|t}^i$ using the FOKF-Standard framework. The prediction step uses the CV motion model (A_t) to determine $X_{t+1|t}^i$. Once the weight estimation state vectors have been obtained, they are utilized as the final tracking trajectory and fed back to the Tracker Ensemble to synchronize the intermediate states of the baseline trackers, thus enabling convergence of the tracking results.

$$X_t^i = [x_c, y_c, w, h, v_x, v_y, v_w, v_h]^T \tag{3.29}$$

$$A_t = \begin{bmatrix} 1 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

where $[x_c, y_c, w, h]$ are same as the above definition of Z_t^i, while $[v_x, v_y, v_w, v_h]$ are the corresponding velocities. As for TLF-FOKF-Standard, I assume all KF trackers have same the state transition matrix A_t, the state-to-measurement matrix H_t^i, the process covariance matrix Q_t and the measurement covariance matrix R_t^i.

Feedback: In order to feed back the fusion results to the baseline trackers, it is necessary to have a thorough understanding of the baseline trackers' algorithms. This is because most baseline trackers were not designed to accommodate feedback. To overcome this issue, it may be necessary to customize the method of feeding back the weight states. This can be achieved by fine-tuning the type of feedback that is provided, such as providing feedback on only the object's location, both the location and scale or the state covariance at a specific time step (either t or $t+1$). The specific type of feedback used should be tailored to the baseline trackers' algorithms and the tracking performance the following feedback.

3.4.2. Analysis

The key advantages of this framework are summarized as follows:

1. **Simple and Fast**: TLF-FOKF-Standard is a simplified and fast algorithm, employing several concise and explicit equations with low time complexity $O(NM)$.

2. **Optimal**: TLF-FOKF-Standard is an optimal TLF algorithm as long as it meets the prerequisites of FOKF-Standard.

3. **Flexible**: TLF-FOKF-Standard has the ability to adapt to any number of baseline trackers based on the flexible design of the values in FOKF-Standard. The most important component of TLF-FOKF-Standard is the Result Fusion, which provides ample room for improvement in order to achieve better tracking performance. For instance, an algorithm for removing baseline trackers could be added to enhance the Tracker Evaluation component and alleviate the requirement for feedback, or an algorithm could be designed to determine the hyperparameters for better approximating the ground truth model.

4. **Explainable**: TLF-FOKF-Standard is a white-box model, meaning that its behaviors, production of fusion results, and the influence of variables can be clearly explained. This is particularly important in actual applications, such as autonomous driving, where minimizing uncontrollable risk is crucial. This is especially relevant at present, where AI models are becoming more intricate and harder to explain.

On the contrary, the drawbacks of TLF-FOKF-Standard are listed as follows:

1. **Similar Measurement Uncertainty**: The performance of FOKF-Standard is best suited for baseline trackers with similar measurement uncertainties (R_t^i) due to the weight estimation design. When the measurement uncertainties significantly differ, it could hinder the performance of FOKF-Standard.

2. **Hyperparameter**: Currently, the values of hyperparameters such as A_t, Q_t, and R_t^i are handcrafted, and may not optimally represent the true model under different scenes.

3. **Baseline Tracker Removal**: The algorithm lacks the ability to remove unreliable measurements, making it sensitive to poor measurements, and necessitating the inclusion of the Feedback component.

3.5. Dissimilar Measurement Uncertainties-Based FOKF

This section focuses on addressing the limitation of FOKF-Standard that requires baseline trackers with similar measurement uncertainties R_t^i. By addressing this limitation, the scope of the FOKF algorithm becomes more flexible and gains wider applicability. For the sake of simplicity, the augmented version of FOKF designed to handle differences in measurement uncertainties will be referred to as FOKF-R. Any symbols used in FOKF-R that are not specifically defined can be taken as the same as those used in Section 3.3.

3.5.1. Problem Formulation

To address the limitation of requiring similar measurement uncertainties and to accommodate the fact that FOKF-Standard utilizes average weighting for the state estimation vectors, I propose FOKF-R. FOKF-R overcomes this limitation by replacing $X_{t|t-1}^w$ with $X_{t|t-1}^i$ to maintain the independence of the state estimation vectors for each KF filter s_i. The rationale behind FOKF's use of average weighting as described in Zhong and Liu, 2021 is mainly to conserve sensor energy by keeping only the most recent state estimation. However, this point is not a major concern for VOT tasks. Therefore, it is feasible to replace $X_{t|t-1}^w$ with $X_{t|t-1}^i$ in FOKF-R.

In terms of the system and the measurement equations, they are as same as Eq.3.1 and 3.2, respectively. The designed FOKF-R filter is shown as follows:

$$e_t^i = a_t^i(Z_t^i - H_t^i X_{t|t-1}^i) \tag{3.31}$$

$$e_t^j = a_t^j(Z_t^j - a_t^j H_t^j X_{t|t-1}^i) \tag{3.32}$$

$$X_{t|t}^i = (c_t^i + d_t^i)X_{t|t-1}^i + a_t^i L_t^i e_t^i + N_t^i \sum_{j \in C_t^i} e_t^j \tag{3.33}$$

$$X_{t+1|t}^i = A_t X_{t|t}^i \tag{3.34}$$

3.5.2. Derivation

Based on Eqs.3.1, 3.33, 3.34, the corresponding equations are the same as Eqs.3.8, 3.9, 3.11. The update estimation error is derived according to Eqs.3.2, 3.31-3.33, which considers $n_t^i = \sum_{j \in C_t^i} a_t^j$, $H_t^i = H_t^j$ and $c_t^i + d_t^i \equiv 1$ for each s_i:

$$\widetilde{X}_{t|t}^i = [I_p - a_t^i L_t^i H_t^i - n_t^i N_t^i H_t^i]\widetilde{X}_{t|t-1}^i - a_t^i L_t^i v_t^i - N_t^i \sum_{j \in C_t^i} a_t^j v_t^j \tag{3.35}$$

The cost function for the optimal estimation is the same as Eq.3.13. Similarly, the prediction error edge-covariance and the prediction error variance are derived as the same as Eqs.3.14 and 3.15.

As for the update error covariance, it can be obtained according to Eqs.3.8 and 3.35 as follows:

$$\begin{aligned} P_{t|t}^{i,j} &\equiv E[\widetilde{X}_{t|t}^i (\widetilde{X}_{t|t}^j)^T] \\ &= [F_t^i P_{t|t-1}^i (F_t^j)^T] + a_t^i a_t^j L_t^i R_t^{i,j} (L_t^j)^T + N_t^i \sum_{r \in C_t^i \cap C_t^j} a_t^r R_t^{r,r} (N_t^j)^T \end{aligned} \tag{3.36}$$

Assume $i = j$, Eq.3.36 is rewritten as follows:

$$P_{t|t}^{i,i} \equiv P_{t|t}^i = [F_t^i P_{t|t-1}^i (F_t^i)^T] + a_t^i L_t^i R_t^i (L_t^i)^T + N_t^i \sum_{j \in C_t^i} a_t^j R_t^j (N_t^i)^T \tag{3.37}$$

Definition 2: A class of distributed KF with the flexible values, the local Kalman gain, and the community Kalman gain but without the weight estimation is defined as the Different Measurement Uncertainties-Based FOKF (FOKF-R), which minimizes the trace $\sum_{i \in T_t} tr\{P_{t|t}^i\}$ and consists of the following Eqs.3.31-3.34, 3.15, and 3.37-3.40.

Theorem 2: According to Eqs.3.1, 3.2, 3.37 and Assumption 1, there exists the following local and community Kalman gains:

$$L_t^i = [a_t^i (c_t^i + d_t^i) P_{t|t-1}^i (H_t^i)^T (I_m - (n_t^i)^2 G_t^i S_t^i) \bar{S}_t^i][I_m - a_t^i (n_t^i)^2 S_t^i G_t^i S_t^i \bar{S}_t^i]^{-1} \tag{3.38}$$

$$N_t^i = [n_t^i (c_t^i + d_t^i) P_{t|t-1}^i (H_t^i)^T (I_m - a_t^i \bar{S}_t^i S_t^i) G_t^i][I_m - a_t^i (n_t^i)^2 S_t^i \bar{S}_t^i S_t^i G_t^i]^{-1} \tag{3.39}$$

where

$$\begin{cases} S_t^i = H_t^i P_{t|t-1}^i (H_t^i)^T \\ \bar{S}_t^i = [a_t^i (S_t^i + R_t^i)]^{-1} \\ G_t^i = [(n_t^i)^2 S_t^i + \sum_{j \in C_t^i} a_t^j R_t^j]^{-1} \end{cases} \tag{3.40}$$

Proof 2: The proof is omitted here, please refer to Proof 1.

3.5.3. Analysis

The analysis of FOKF-R closely resembles that of FOKF-Standard, with the exception being that the weight step is omitted to avoid the issue of needing similar measurement uncertainties. In terms of prerequisites for achieving optimal performance, FOKF-R is similar to FOKF-Standard, with the exception that it does not require $R_t^i = R_t$ as long as the accurate value for R_t^i is known. Additional experiments are conducted in Sections 4.1.2-4.1.5 to provide a more extensive evaluation of FOKF-R.

3.6. Tracker-Level Fusion Based on FOKF-R

This section presents a new approach, TLF-FOKF-R, for addressing the VOT task. This method replaces FOKF-Standard with FOKF-R and is described in detail, along with its key differences from TLF-FOKF-Standard, in the following paragraphs.

As shown in Fig.3.4, TLF-FOKF-R is structurally similar to TLF-FOKF-Standard, but features a novel definition of the fusion result, $X_{t|t}^w$ and $P_{t|t}^w$, where $w = \{i \in T_t : min(tr(P_{t|t}^i))\}$. Because $P_{t|t}^i$ is

Algorithm 2: FOKF-R for object tracking

Initialize: $\bar{X}_{0|-1} = X_0$, $\bar{P}_{0|-1} = P_0$, set $T_t = \{\emptyset\}$ and $C_t^i = \{\emptyset\}$ if they will change, $a_t^i = c_t^i = d_t^i = n_t^i = 0$.
Input: Measurement vectors Z_t^i.
1: Update T_t and C_t^i if they change and update the flexible values (a_t^i, c_t^i, d_t^i) and n_t^i based on measurements.
2: Do the update step of FOKF-R by using Eqs.3.31-3.33, 3.37-3.40.
3: Do the prediction step of FOKF-R by using Eqs.3.34 and 3.15.
4: Update time step t, jump to *Step* 1.

the update state covariance matrix which can represent the distance between the update estimation $X_{t|t}^i$ and the true state vector X_t, I select the $X_{t|t}^i$ with the minimal $P_{t|t}^i$ among all s_i as the fusion result and feed it back to Tracker Ensemble according to the trace of $P_{t|t}^i$. It makes up for the lack of the weight step of FOKF-R to provide the fusion result. The key advantages of TLF-FOKF-R are the same as TLF-FOKF-Standard, while it eliminates the drawback of similar measurement uncertainty. In addition, TLF-FOKF-R can possess a part of the function of the baseline tracker removal algorithm if R_t^i is accurate. The rest content of this section is similar to that in Section 3.4.

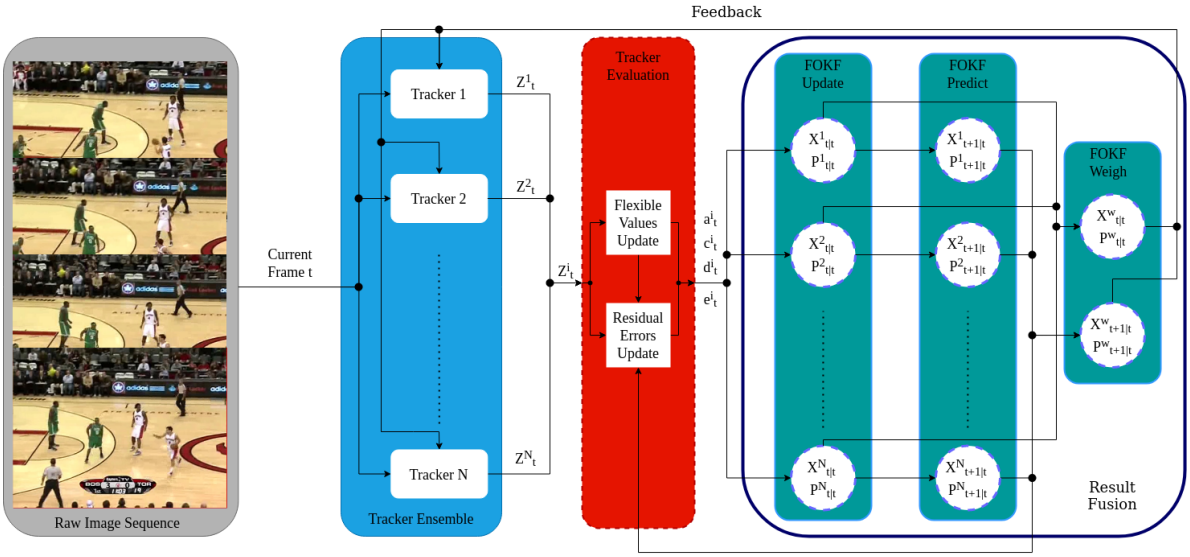


Figure 3.4: The main framework of TLF-FOKF-R: it is similar with TLF-FOKF-Standard, excluding FOKF Weigh and Residual Update. As for FOKF Weigh, it is used for the fusion result where $w = \{i \in T_t : min(tr(P_{t|t}^i))\}$. In terms of residual error update, it follows the definition of FOKF-R.

3.7. Main Baseline Tracker-Based FOKF

In this section, I propose the Main Baseline Tracker-Based FOKF (FOKF-M) in detail. In the context of the VOT task, it's observed that TLF methods only need one fusion result as the final outcome. Inspired by FOKF-Standard and FOKF-R, I note that $X_{t|t}^i$ is weighted by $L_t^i e_t^i$ and $N_t^i \sum_{j \in C_t^i} e_t^j$ where the values of L_t^i and N_t^i determines whether s_i relies more on the local measurement e_t^i or the sum of the community measurements e_t^j. But when the number of baseline trackers is large, the weight of a single e_t^j gets diluted by the step of summing and averaging. In this way, e_t^i plays a more important role than that of individual e_t^j. Based on this observation, Table 3.2 shows that FOKF-Standard performs best when the real R_t^i is similar, while FOKF-R performs best when the estimated R_t^i is accurate enough for the real R_t^i. However, such ideal conditions are less common in engineering applications, where it's challenging to ensure that the actual R_t^i is consistently similar or accurate estimated R_t^i can be obtained. The most possible situation is that we only know which baseline tracker performs best according to experience but without knowing the accurate real R_t^i. Furthermore, considering the computational complexity,

especially when the number of baseline trackers is substantial, FOKF-M is introduced to address these scenarios.

Table 3.2: A comparison of FOKF-Standard, FOKF-R, and FOKF-M under different prerequisites when the baseline tracker's number is large. The more +, the better performance. Here, 'partially' denotes at least knowing which baseline tracker is the best.

Similar real R_t^i	Accurate estimated R_t^i	FOKF-Standard	FOKF-R	FOKF-M
Yes	No	+++	++	+
No	Yes	++	+++	++
No	Partially	+	++	+++

The essence of FOKF-M is to leverage two types of prior knowledge: first, that TLF necessitates only one fusion result, and second, that we know which baseline tracker performs the best among several baseline trackers. In contrast to FOKF-R, FOKF-M defines the best baseline tracker as the primary baseline tracker m, with its corresponding KF tracker s_m as the only KF tracker. As a result, the time complexity of FOKF-M is reduced to $O(M)$.

3.7.1. Problem Formulation

As for the system and the measurement equations, they are the same as Eq.3.1 and 3.2. Based on the aforementioned discussion, there is only one KF tracker s_m and the designed FOKF-M filter is shown as follows:

$$e_t^m = a_t^m(Z_t^m - H_t^m X_{t|t-1}^m) \tag{3.41}$$

$$e_t^j = a_t^j(Z_t^j - a_t^j H_t^j X_{t|t-1}^m) \tag{3.42}$$

$$X_{t|t}^m = (c_t^m + d_t^m)X_{t|t-1}^m + a_t^m L_t^m e_t^m + N_t^m \sum_{j \in C_t^m} e_t^j \tag{3.43}$$

$$X_{t+1|t}^m = A_t X_{t|t}^m \tag{3.44}$$

3.7.2. Derivation

Based on Eqs.3.1, 3.43, 3.44, the corresponding equations are defined as:

$$\widetilde{X}_{t+1|t}^m \equiv (c_t^m + d_t^m)X_{t+1} - X_{t+1|t}^m \tag{3.45}$$

$$\widetilde{X}_{t|t}^m \equiv (c_t^m + d_t^m)X_t - X_{t|t}^m \tag{3.46}$$

Based on Eqs.3.1, 3.44, 3.45, it can derive as follows:

$$\widetilde{X}_{t+1|t}^m = A_t \widetilde{X}_{t|t}^m + (c_t^m + d_t^m)w_t \tag{3.47}$$

For Eqs.3.2, 3.41-3.43 and 3.45-3.46, the update estimation error is derived as follows:

$$\widetilde{X}_{t|t}^m = [(c_t^m + d_t^m)I_p - a_t^m L_t^m H_t^m - n_t^m N_t^m H_t^m]\widetilde{X}_{t|t-1} - a_t^m L_t^m v_t^m - N_t^m \sum_{j \in C_t^m} a_t^j v_t^j \tag{3.48}$$

where $n_t^m = \sum_{j \in C_t^m} a_t^j$ and I assume $H_t^m = H_t^j$.

The optimal estimation can be obtained by minimizing the following cost function:

$$J = E[\widetilde{X}_{t|t}^m (\widetilde{X}_{t|t}^m)^T] \tag{3.49}$$

According to Eqs.3.45 and 3.49, the prediction error variance is defined and derived as follows:

$$P_{t+1|t}^m \equiv E[\widetilde{X}_{t+1|t}^m (\widetilde{X}_{t+1|t}^m)^T] = A_t P_{t|t}^m (A_t)^T + (c_t^m + d_t^m)Q_t \tag{3.50}$$

Similarly, the update error variance can be obtained according to Eq.3.48:

$$\begin{aligned} P_{t|t}^m &\equiv E[\widetilde{X}_{t|t}^m (\widetilde{X}_{t|t}^m)^T] \\ &= [F_t^m P_{t|t-1}^m (F_t^m)^T] + a_t^m L_t^m R_t^m (L_t^m)^T + N_t^m \sum_{j \in C_t^m} a_t^j R_t^j (N_t^m)^T \end{aligned} \tag{3.51}$$

where $F_t^m = (c_t^m + d_t^m)I_p - a_t^m L_t^m H_t^m - n_t^m N_t^m H_t^m$.

Definition 3: A class of distributed KF with the flexible values, the local Kalman gain, the community Kalman gain, and the main KF tracker is defined as the Main Baseline Tracker-based FOKF (FOKF-M), which minimizes the trace $tr\{P_{t|t}^m\}$ and consists of the following Eqs.3.41-3.44, and 3.50-3.54.

Theorem 3: According to Eqs.3.1, 3.2, 3.51 and Assumption 1, there exists the following local and community Kalman gains:

$$L_t^m = [a_t^m(c_t^m + d_t^m)P_{t|t-1}^m(H_t^m)^T(I_m - (n_t^m)^2 G_t^m S_t^m)\bar{S}_t^m][I_m - a_t^m(n_t^m)^2 S_t^m G_t^m S_t^m \bar{S}_t^m]^{-1} \tag{3.52}$$

$$N_t^m = [n_t^m(c_t^m + d_t^m)P_{t|t-1}^m(H_t^m)^T(I_m - a_t^m \bar{S}_t^m S_t^m)G_t^m][I_m - a_t^m(n_t^m)^2 S_t^m \bar{S}_t^m S_t^m G_t^m]^{-1} \tag{3.53}$$

where

$$\begin{cases} S_t^m = H_t^m P_{t|t-1}^m (H_t^m)^T \\ \bar{S}_t^m = [a_t^m(S_t^m + R_t^m)]^{-1} \\ G_t^m = [(n_t^m)^2 S_t^m + \sum_{j \in C_t^m} a_t^j R_t^j]^{-1} \end{cases} \tag{3.54}$$

Proof 3: The proof is omitted here, please refer to Proof 1.

3.7.3. Analysis

The analysis of FOKF-M is similar to that of FOKF-Standard, except for two differences: first, there's only one KF tracker, and second, the weight step is removed in FOKF-M. The prerequisites for FOKF-M's optimal performance are also similar to those of FOKF-R. Further details on performance evaluations can be found in Sections 4.1.2-4.1.5, where we report the results of extensive experiments.

Algorithm 3: FOKF-M for object tracking

Initialize: $\bar{X}_{0|-1} = X_0$, $\bar{P}_{0|-1} = P_0$, set $T_t = \{\emptyset\}$ and $C_t^m = \{\emptyset\}$ if they will change, $a_t^i = c_t^m = d_t^m = n_t^m = 0$.
Input: Measurement vectors Z_t^i.
1: Update T_t and C_t^m if they change and update the flexible values (a_t^i, c_t^m, d_t^m) and n_t^m based on measurements.
2: Do the update step of FOKF-M by using Eqs.3.41-3.43, 3.51-3.54.
3: Do the prediction step of FOKF-M by using Eqs.3.44 and 3.50.
4: Update time step t, jump to *Step* 1.

3.8. Tracker-Level Fusion Based on FOKF-M

This section introduces TLF-FOKF-M for the VOT task by replacing FOKF-R with FOKF-M. It outlines the framework of TLF-FOKF-M and highlights the differences between TLF-FOKF-R and TLF-FOKF-M. As depicted in Fig.3.5, TLF-FOKF-M emphasizes the use of a single KF tracker s_m to handle the measurements from the baseline trackers. Consequently, the estimation $X_{t|t}^m$ of the corresponding KF tracker serves as the fusion result. The estimation states are then fed back to the Tracker Ensemble and residual error update module for further iteration. TLF-FOKF-M delivers the same benefits as TLF-FOKF-R while reducing the need for precise knowledge of R_t^i and focusing on a priori knowledge of the best baseline tracker. Additionally, TLF-FOKF-M reduces the time complexity from $O(NM)$ to $O(M)$. The remaining content of this section is similar to Section 3.4.

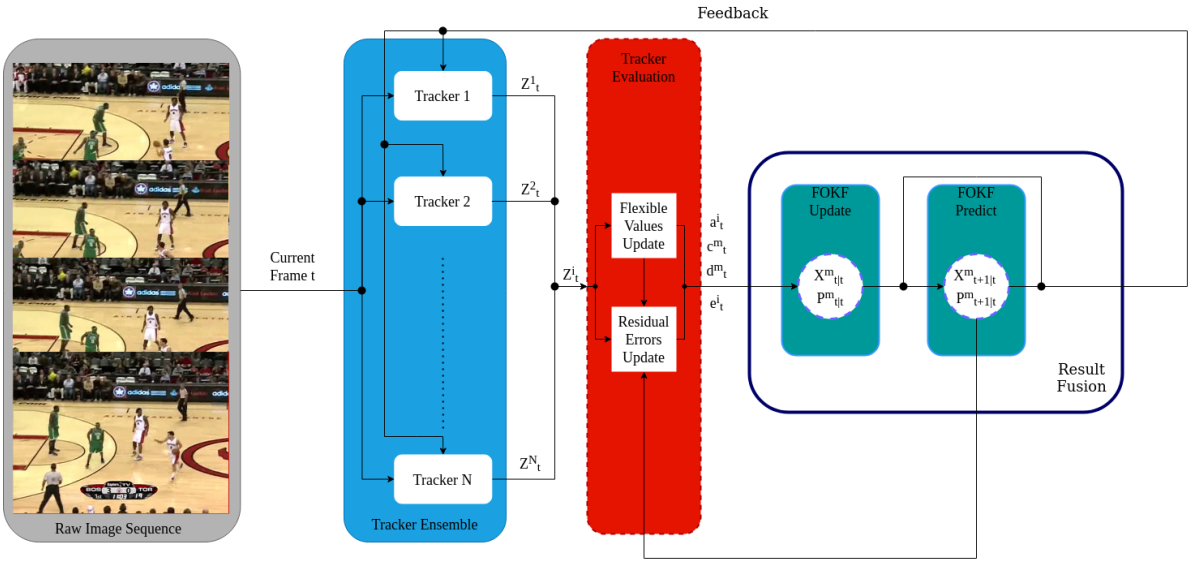


Figure 3.5: The framework of TLF-FOKF-M. The key differences between TLF-FOKF-R and TLF-FOKF-M are that there is only one KF tracker and the weight step is deleted in TLF-FOKF-M.

4

Experiments

This chapter reports two primary experiments. The first experiment aims to demonstrate the efficiency of FOKF-Standard, FOKF-R, and FOKF-M in trajectory estimation tasks under ideal circumstances. To simplify matters, FOKF-Standard is referred to as FOKF-S. The second experiment evaluates the VOT tracking performance enhanced by TLF-FOKF-S, TLF-FOKF-R, and TLF-FOKF-M. These experiments also support my in-depth analysis of the impact of baseline trackers in Section 3.2.

4.1. FOKF-Related Experiments

There are two reasons why it is crucial to evaluate the effectiveness of FOKF under perfectly modeled conditions. Firstly, FOKF forms the basis of TLF-FOKF. Secondly, it is hard to accurately model the VOT scenario of TLF-FOKF at present, which could introduce uncertainty in the results. For example, it is challenging to determine if the tracking results of the baseline trackers comply with a Gaussian distribution and estimate hyperparameters accurately. Therefore, this section aims to assess the effectiveness of FOKF-S, FOKF-R, and FOKF-M under various ideal conditions, focusing on the critical role of measurement uncertainties. These ideal conditions include scenarios where measurement uncertainties are uniform and accurately known.

4.1.1. Implementation Details

To demonstrate the performance of the trajectory estimation of FOKF-related (FOKF-S, FOKF-R, and FOKF-M), I design a monitored area, where the KF tracker s_i can measure the target and transfer the messages to all other KF trackers in this area. The area could be measured in meters or pixels, with the latter demonstrating no impact on performance evaluation. To decrease the complexity of testing, I assume there is no update for T_t and C_t^i. To decrease the effect caused by the random motion disturbance w_t of the system for the experiment results, I run 50 times simulation with identical parameters (A_t, Q_t, and H_t^i) to obtain the mean estimation error and the estimation error standard derivation (std).

Moreover, the initial values for the target and each s_i are defined as $X_0 = [0 \quad 0]^T$, $X_{0|0}^i = [-10 + 20 * rand \quad -10 + 20 * rand]^T$, $P_{0|0}^i = 10Ip$, where $rand$ denotes a random function that generates uniformly distributed random number between 0 and 1. Here, the estimated state vector $X_{t|t}^i$ represents the horizontal and vertical position of this area. And the total time steps $t = 500$. The weight estimation of FOKF-R uses the definition in Section 3.6. As for FOKF-M, the selection of s_m is the first KF tracker s_1 for simplicity. In addition, the typical KF (Kalman, 1960) is also tested to clearly show the improvement of FOKF-related, where it obtains the measurement Z_t^1 for iteration. As for the systems, the parameters

are set as follows:

$$A_t = \begin{bmatrix} 0.9996 & -0.0300 \\ 0.0300 & 0.9996 \end{bmatrix} \tag{4.1}$$

$$Q_t = \begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix} \tag{4.2}$$

$$H_t^i = \begin{bmatrix} 1 & 1 \end{bmatrix} \tag{4.3}$$

For a real VOT task, the real R_t^i of the baseline tracker is unknown, and the only way is to estimate it. Therefore, FOKF-related experiments divide the R_t^i as two parts for testing, where the real R_t^i is for generating the measurement (Eq.3.2) while the estimated R_t^i is used for KF tracker s_i to estimate uncertainty, such as Eq.3.18). As for the evaluation metrics, the mean estimation error and the total mean estimation error (tae) are defined as follows:

$$er_t = \sqrt{(X_{1,t} - X_{1,t|t}^w)^2 + (X_{2,t} - X_{2,t|t}^w)^2} \tag{4.4}$$

$$tae = \frac{1}{steps} \sum_{t=1}^{steps} er_t \tag{4.5}$$

where $X_{t|t}^w$ is the final result of each method as mentioned in Chapter 3.

4.1.2. Experiment 1: Known and Similar Measurement Uncertainties

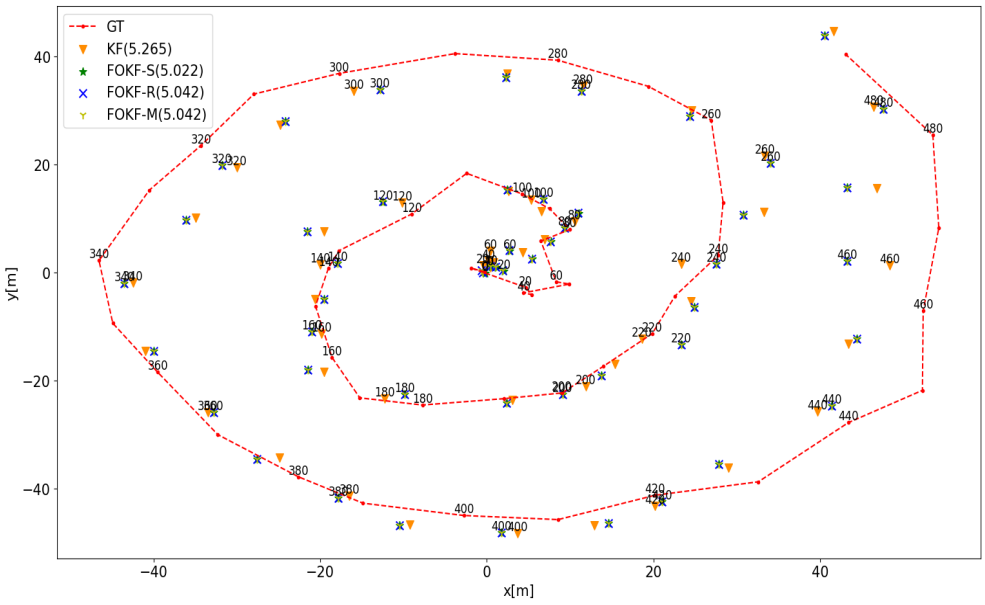


Figure 4.1: Experiment 1: Snapshot of the estimation of KF, FOKF-S, FOKF-R, FOKF-M, and the real position of GT when the tracker number is 10 and the measurement uncertainties are known and similar. It is a snapshot of one of the 50 times simulations that the final estimated positions and the real positions are given every 10-time steps, whereas the time steps are given every 20-time steps. For aesthetics, the shown GT points are connected to represent truth trajectories. The numbers after the legends are the tae.

This experiment aims at testing the performance of FOKF-related when $R_t^i = R_t = 50$ are set both for the real R_t^i and the estimated R_t^i. A qualitative example is shown in Fig.4.1. With regard to the effect of the tracker number for FOKF-related, Fig.4.2 shows the performance under different tracker numbers from 1 to 20 of the typical KF and FOKF-related. Both of these two Figures can clearly see

that FOKF-related (mean estimation error around $4.352m$ and std 0.715) significantly outperforms the typical KF (mean estimation error $4.889m$ and std 0.734), which shows FOKF-related take advantage of collaborative information to improve tracking accuracy and robustness. Moreover, the obvious trend that the performance is improved with the increase of the tracker number can be seen in Fig.4.2. Furthermore, FOKF-S slightly surpasses FOKF-R and FOKF-M with an overall $0.001m$, while the largest boost is $0.007m$ when the tracker number is 9. The above experiments show that FOKF-S is the best among FOKF-related under the situation of known and similar R_t^i, while FOKF-R and FOKF-M have the same performance because they have the same R_t^i and select the estimation result of the same KF tracker s_1 as the final result.

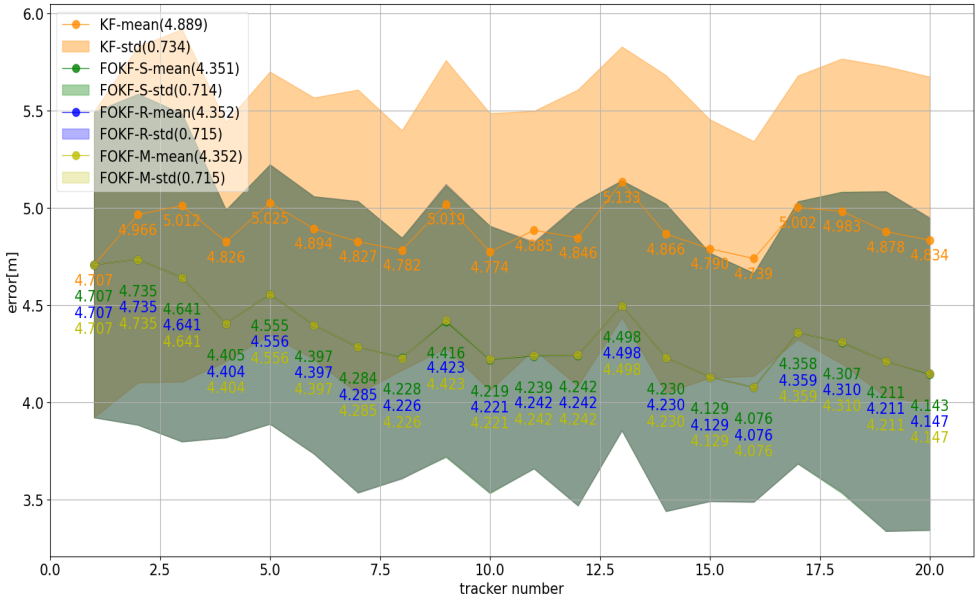


Figure 4.2: Experiment 1: Performance influence on tracker number of the Tracker Ensemble of FOKF-S, FOKF-R, FOKF-M when the measurement uncertainties are known and similar. The numbers after the legends are the average tae and its standard derivation of different tracker numbers, while the numbers near the point are the average tae of 50 times simulation.

4.1.3. Experiment 2: Unknown and Similar Measurement Uncertainties

This experiment aims at testing the performance of FOKF-related when the real $R_t^i = R_t = 50$ for Eq.3.2 are unknown for KF tracker s_i, where the inaccurate estimated $R_t^i = 50 + 20 * (i-1)$ are estimated. Similar to Section 4.1.2, Figs.4.3 and 4.4 are shown as follows. Similar to that in Section 4.1.2, FOKF-related significantly outperforms the typical KF, which is shown in Fig.4.4. Also, the accuracy of FOKF-S ($4.399m$) is better than FOKF-R and FOKF-M ($4.415m$). Note that s_1 has the overall lowest uncertainty measurements, which is satisfied with the requirement of FOKF-M knowing the best KF tracker.

4.1.4. Experiment 3: Known and Dissimilar Measurement Uncertainties

This experiment aims at testing the performance of FOKF-related when the dissimilar real $R_t^i = 50 + 20 * (i-1) * rand_1 - 40 * rand_2$ are known for KF tracker s_i. Here, $rand$ is updated for each KF tracker s_i at each tracker number. Different from Section 4.1.3, FOKF-R is the best method ($3.374m$) when the real R_t^i is dissimilar and known because it is not limited by the average step of FOKF-S or the simple single KF tracker fusion estimation of FOKF-M.

4.1.5. Experiment 4: Unknown and Dissimilar Measurement Uncertainties

This experiment aims at testing the performance of FOKF-related when the dissimilar real $R_t^i = 50 + 20 * (i-1) * rand_1 - 40 * rand_2$ are unknown for KF tracker s_i, where the inaccurate estimated

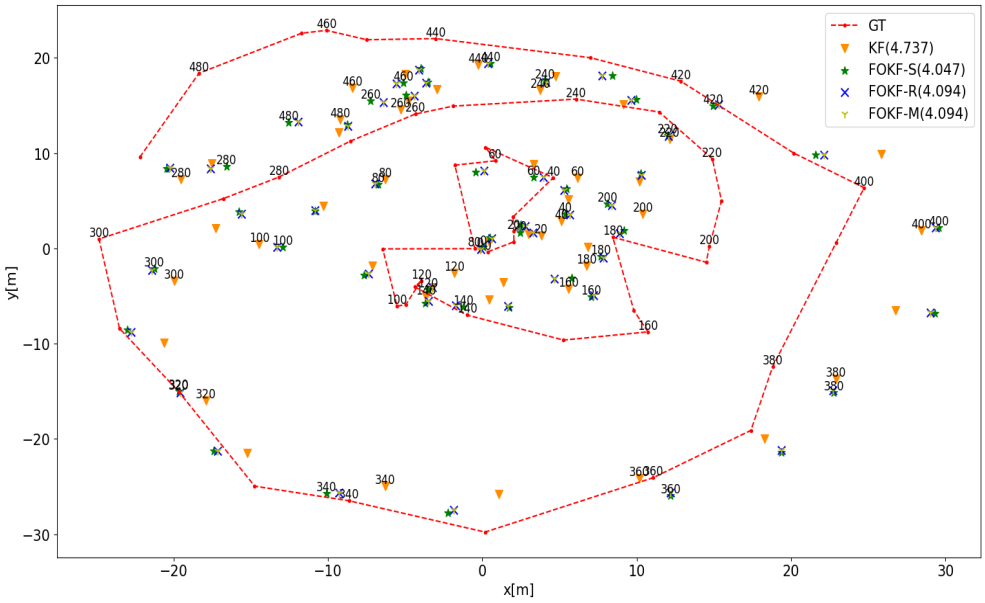


Figure 4.3: Experiment 2: Snapshot when the measurement uncertainties are unknown and similar. The rest is similar to the caption of Fig.4.1.

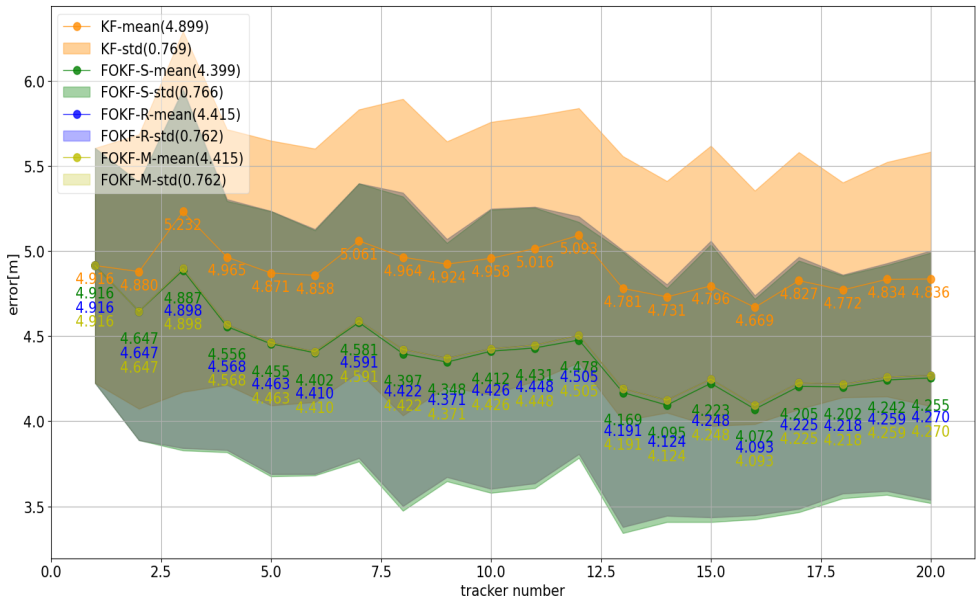


Figure 4.4: Experiment 2: Tracker number influence when the measurement uncertainties are unknown and similar. The rest is similar to the caption of Fig.4.1.

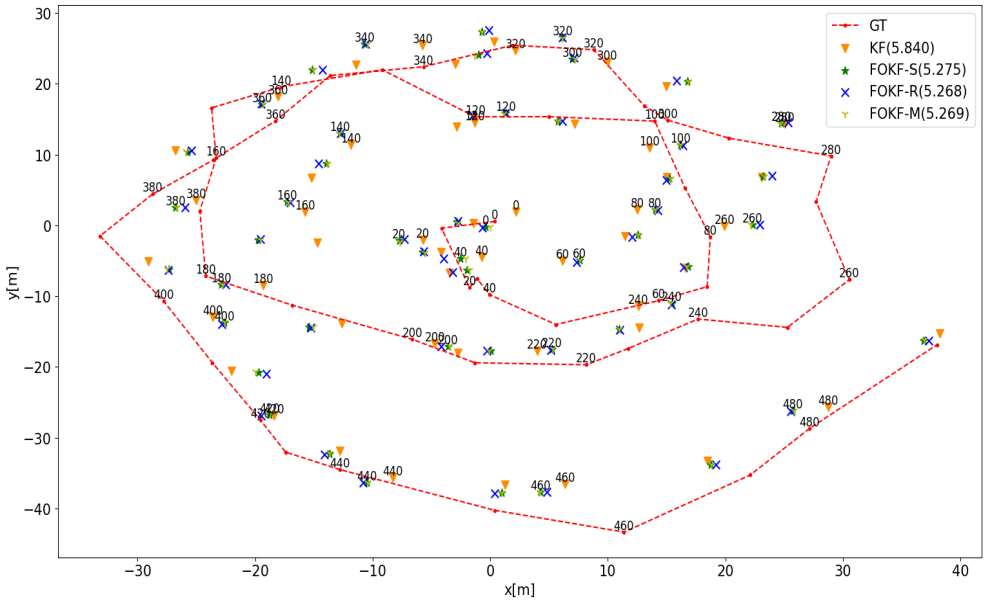


Figure 4.5: Experiment 3: Snapshot when the measurement uncertainties are known and dissimilar. The rest is similar to the caption of Fig.4.1.

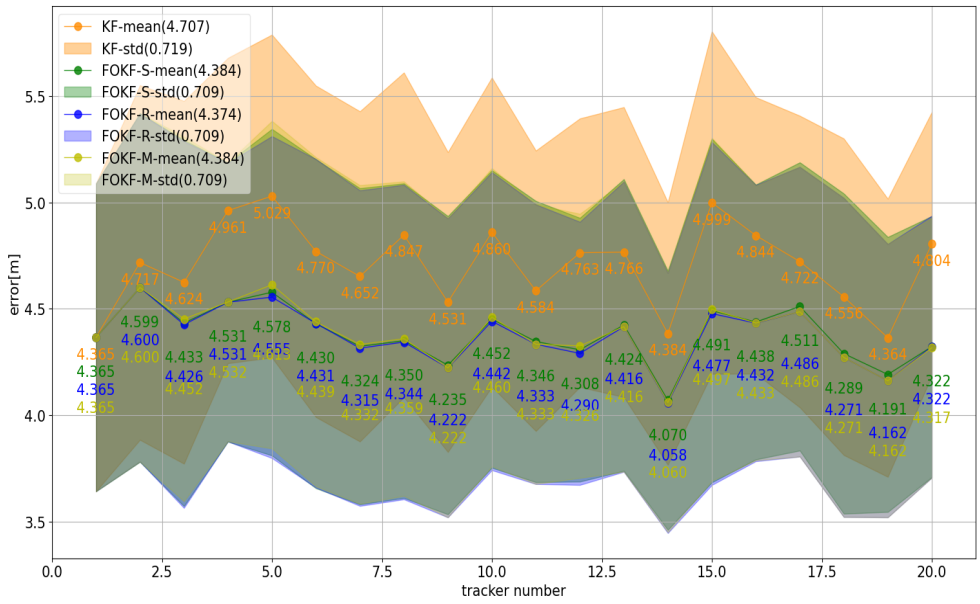


Figure 4.6: Experiment 3: Tracker number influence when the measurement uncertainties are known and dissimilar. The rest is similar to the caption of Fig.4.1.

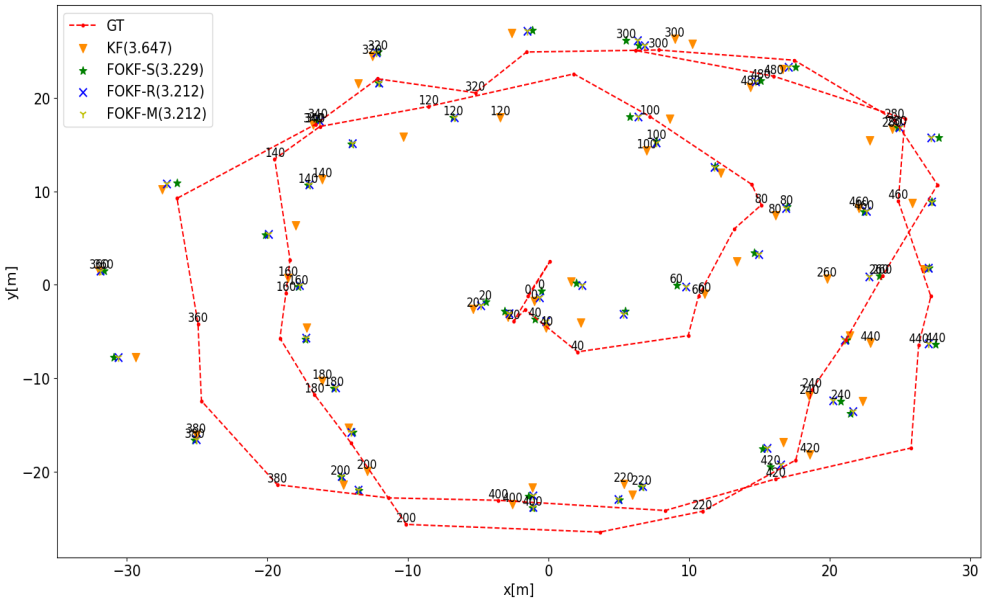


Figure 4.7: Experiment 4: Snapshot when the measurement uncertainties are unknown and dissimilar. The rest is similar to the caption of Fig.4.1.

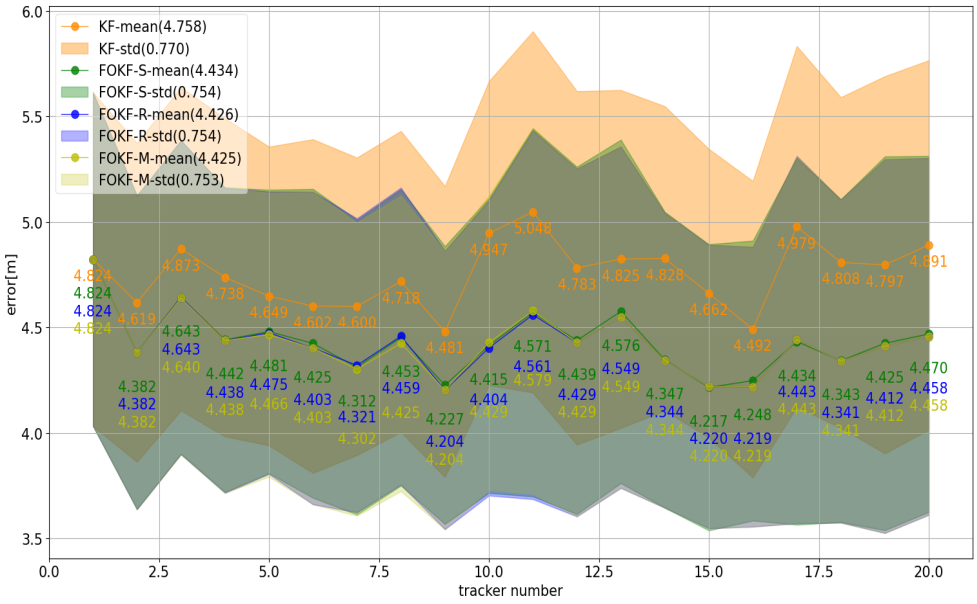


Figure 4.8: Experiment 4: Tracker number influence when the measurement uncertainties are unknown and dissimilar. The rest is similar to the caption of Fig.4.1.

$R_t^i = 50 + 40 * (i-1) * rand_3$ are estimated for trajectory estimation. As shown in Figs.4.7 and 4.8, FOKF-M is the best method in this situation because the selection of the main tracker can weaken the adverse conditions to some extent that R_t^i is dissimilar and unknown, where the former hinders FOKF-S while the latter inhibits FOKF-R.

4.1.6. Discussion

Based on the results of the experiments, we can validate Table 3.2. In terms of fusion performance, FOKF-related algorithms have distinct advantages and drawbacks under different circumstances. FOKF-S performs better when the baseline trackers have similar R_t^i. FOKF-R performs better when accurate estimation of R_t^i is possible, even if R_t^i values are dissimilar. FOKF-M is competitive when the best baseline tracker is known, without facing the dilemma of FOKF-R's need for accurate R_t^i. In terms of time complexity, FOKF-M is the fastest method, with a complexity of $O(M)$. FOKF-S and FOKF-R have the same time complexity of $O(NM)$. These experiments were conducted on a laptop with an AMD Ryzen 7 4800H CPU. For instance, when the number of trackers is 2, the processing times for the typical KF, FOKF-S, FOKF-R, and FOKF-M are $0.1ms$, $1.3ms$, $1.3ms$, and $0.6ms$, respectively. When the number of trackers is 10, the processing times for FOKF-S, FOKF-R, and FOKF-M are $7.2ms$, $7.2ms$, and $0.9ms$, respectively.

Section 4.1 primarily evaluates the performance of FOKF-related algorithms under various R_t^i situations, with a particular focus on scenarios where R_t^i values are known and similar for KF tracker s_i. This is an important consideration since estimating R_t^i for baseline trackers in VOT tasks is often challenging. The experimental results demonstrate that FOKF-related algorithms are highly effective in leveraging collaborative information to improve tracking performance, as demonstrated by their superior performance over the typical KF. Additionally, the experiments reveal that FOKF-S, FOKF-R, and FOKF-M exhibit different fusion performances under different R_t^i situations, providing valuable insights for selecting the most appropriate FOKF-related method for VOT tasks.

4.2. TLF-FOKF-Related Experiments

This section focuses on testing TLF-FOKF-related methods on three challenging VOT benchmarks. TLF-FOKF-related includes TLF-FOKF-Standard (namely TLF-FOKF-S for simplicity), TLF-FOKF-R, TLF-FOKF-M. First, implementation details and descriptions of testing benchmarks are provided. To demonstrate the effectiveness of TLF-FOKF-related methods, comparative experiments with baseline trackers are conducted by considering similar measurement uncertainties. Additionally, to investigate the positive effect of tracker diversity and the negative effect of acceptability of baseline tracker for feedback, experiments with two baseline trackers are designed with similar measurement uncertainties to eliminate interference from other factors. The influence of the number of baseline trackers for TLF-FOKF-related and their corresponding running speeds are also evaluated, which also addresses the negative effect of a limited number of baseline trackers. Dissimilar measurement uncertainties are used to test the sensitivity of TLF-FOKF-related and to support the claim of the negative effects of occasional and inevitable poor tracking results. Experiments to analyze the influence of process uncertainty are also conducted. Due to differences in baseline trackers and a scarcity of public repositories of TLF methods, comparisons with state-of-the-art TLF methods are based on inference. Finally, the strengths and weaknesses of TLF-FOKF-related are analyzed based on the experiment results, and the implications and potential impact of the findings are discussed.

4.2.1. Implementation Details

The TLF-FOKF-related methods were implemented using Python 3.8 and PyTorch 1.7.0. The experiments were conducted on a laptop equipped with an AMD Ryzen 7 4800H CPU and a 4GB GeForce GTX 1650 GPU.

Hyperparameters: As for TLF-FOKF-related methods, there are 6 hyperparameters needed to be determined in advance, including X_t, Z_t^i, A_t, Q_t, H_t^i and R_t^i. The state vector X_t^i and the measurement Z_t^i which are defined as Eqs.3.28 and 3.29 are based on the requirement of the VOT task. And then the state-to-measurement matrix H_t^i is derived according to X_t^i and Z_t^i as Eq.4.6. The rest hyperparameters are difficult but important to be estimated accurately. But because the goal of this thesis is not focusing on estimating A_t, Q_t, and R_t^i, I simply handcraft them roughly. The state transition matrix A_t is derived as Eq.3.30 according to X_t^i. Motivated by Wojke et al. (2017), the uncertainty is relevant with the std of

position and velocity and the height h of the bbox of the target. The std of position and velocity are set as $1/20$ and $1/160$, respectively. Thus, I define that $\sigma_{pos} = std_{pos} * h$ and $\sigma_{vel} = std_{vel} * h$. And then, the processing Gaussian covariance matrix Q_t and the measurement Gaussian covariance matrix are defined as Eqs.4.7 and 4.8, respectively. These hyperparameters are validated on OTBCOCO which is described in Section 4.9.

$$H_t^i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.6}$$

$$Q_t = \begin{bmatrix} 100\sigma_{pos}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100\sigma_{pos}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100\sigma_{pos}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100\sigma_{pos}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100\sigma_{vel}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100\sigma_{vel}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 100\sigma_{vel}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100\sigma_{vel}^2 \end{bmatrix} \tag{4.7}$$

$$R_t = \begin{bmatrix} \sigma_{pos}^2 & 0 & 0 & 0 \\ 0 & \sigma_{pos}^2 & 0 & 0 \\ 0 & 0 & \sigma_{pos}^2 & 0 \\ 0 & 0 & 0 & \sigma_{pos}^2 \end{bmatrix} \tag{4.8}$$

Tracker Ensemble: The Tracker Ensemble is the only part for TLF-FOKF-related requiring training or a pre-trained model. Since training on a benchmark's training set alone is not essential to compare the improvements of TLF-FOKF-related with baseline trackers, all baseline trackers were trained using pre-trained models and pre-defined hyperparameters to save time. The training set of each candidate baseline tracker is shown in Table 4.1. Additionally, the same baseline tracker was tested with different backbones. The baseline trackers evaluated in this study include MOSSE, GOTURN, ECO_HC (Using handcrafted features), SiamRPN, SiamRPN++_ResNet (He et al., 2016), SiamRPN++_AlexNet (Krizhevsky et al., 2017), SiamRPN++_Mobilev2 (Howard et al., 2017), SiamFC++_AlexNet, SiamFC++ _GoogleNet (Szegedy et al., 2015), STMTrack and STARK. Here STARK is the version of STARK _Lightning which is accelerated by ONNXRUNTIME. The citations indicate the works proposing the backbone. Yolov3_DeepSORT was excluded from the evaluation as it is not suitable for the VOT task due to a lack of overlap between the training set of Yolov3 and the test set of VOT benchmarks.

Table 4.1: The training set of some baseline trackers

Baseline Tracker	Training Set
GOTURN	ALOV300++ (Smeulders et al., 2013) + ImageNet (Russakovsky et al., 2015)
SiamRPN	Youtube-BB (Real et al., 2017) + ImageNet
SiamRPN++	COCO (T.-Y. Lin et al., 2014 + ImageNet + Youtube-BB
SiamFC++	COCO + ImageNet + Youtube-BB + LaSOT (Fan et al., 2019) + GOT-10k
STMTrack	TrackingNet + LaSOT + GOT-10k (Huang et al., 2019) + ImageNet + COCO
STARK	TrackingNet (Muller et al., 2018) + LaSOT + GOT-10k + COCO

Feedback: To reduce the impact of outlier baseline trackers, it is essential to provide feedback on the fusion result to each baseline tracker in TLF-FOKF-S, TLF-FOKF-R, and TLF-FOKF-M. The feedback mechanism is described in Table 4.2.

4.2.2. Descriptions of Testing Benchmarks

The performance of TLF-FOKF-related was evaluated on three challenging tracking datasets: OTB (Wu et al., 2015), GOT-10k (Huang et al., 2019), and LaSOT testing set (Fan et al., 2019), after considering the characteristics of each dataset. OTB focuses mainly on mid-term sequences (sequences

Table 4.2: Feedback detail for baseline trackers

Baseline Tracker	Feedback detail
MOSSE	Position
GOTURN	Position + Scale
ECO_HC	Position on OTB while Position + Scale on GOT-10k and LaSOT
SiamRPN	Position on OTB while Position + Scale on GOT-10k and LaSOT
SiamRPN++	Position + Scale
SiamFC++	Position + Scale
STMTrack	Position + Scale on OTB while Position on GOT-10k and LaSOT
STARK	Position + Scale

with more than 100 frames but less than 1000 frames) with a few short and long sequences. GOT-10k comprises short-term sequences (sequences with less than 100 frames), while LaSOT consists of long-term sequences (sequences with more than 1000 frames). This evaluation allowed for a comprehensive assessment of the performance of TLF-FOKF-related. Fig.4.9 shows some examples from these datasets.

OTB: OTB is a popular and widely used dataset, that comprises short, medium, and long sequences of multiple classes of targets on multiple scenes. The total dataset, OTB2015, comprises 100 sequences collected from common object-tracking videos. The precision and success plot are the two main metrics for evaluation. The precision plot represents the percentage of frames where the tracking results are within the location pixel error threshold of IT. The success plot denotes the ratio of successful frames when the overlap threshold varies from 0 to 1. The successful frame means its overlap is larger than the given threshold. The area under the curve (AUC) of the success plot is used to rank tracking algorithms. The AUC scores of the above two plots are used as the performance measure. Moreover, I select 50 sequences from OTB2015 as the validation set to fine-tune hyperparameters according to whether Yolov3 (Redmon and Farhadi, 2018) (trained on the COCO dataset) can detect the objects, which is called OTBCOCO.

GOT-10k: GOT-10k is a novel and challenging dataset, where the sequences are taken from the moving object in the real world. The test set embodies 84 object classes and 32 motion classes with only 180 video sequences, which is also regarded as a short-term dataset because the sequence is usually around 100 frames. The validation set has a similar structure to the test set The main evaluation metrics are the average overlap and the success rate. The average overlap is the mean of IoU in every frame, while the success rate is counted by the percentage of frames where the overlap is larger than the IoU threshold (0.5 in this thesis). And GOT-10k provides an online evaluation server to avoid tuning parameters on it.

LaSOT: LaSOT focuses on the long-term VOT task with a large scale and high quality. Its test set contains 70 classes target where each class has 4 sequences and an average sequences length of around 2,500 frames. For simplicity, I select the AUC scores as the same as OTB.

4.2.3. Experiment 1: Comparison with Baseline Trackers

To demonstrate the effectiveness of TLF-FOKF-related methods when the measurement uncertainties are similar, it is important to compare the tracking performance of baseline trackers with their TLF-FOKF-related methods. Table 4.3 presents the experimental results on OTB2015, while Table 4.4 and Table 4.5 show the results on the GOT-10k test set and LaSOT test set, respectively.

From the above tables, it is clear to see the effectiveness of TLF-FOKF-related methods for the improvement of tracking performance as long as the hyperparameters can approximate the real model. For example, TLF-FOKF-related improve around 1.3% AUC scores of precision and success on OTB2015 based on Table 4.3 when there are five baseline trackers. In addition, a similar level of accuracy improvement is shown on the GOT-10k test set according to Table 4.4. The largest accuracy improvement is shown on the LaSOT test set according to the left column of Table 4.5, where the AUC scores are increased by around 6% with only using three baseline trackers. Notice that the right column of Table 4.5 shows that the precision AUC scores of TLF-FOKF-related are slightly smaller than the baseline tracker STMTrack (0.1%) which is caused by the difference of the measurement uncertainties on the precision of these two baseline trackers too large, while the success AUC scores show normally because the

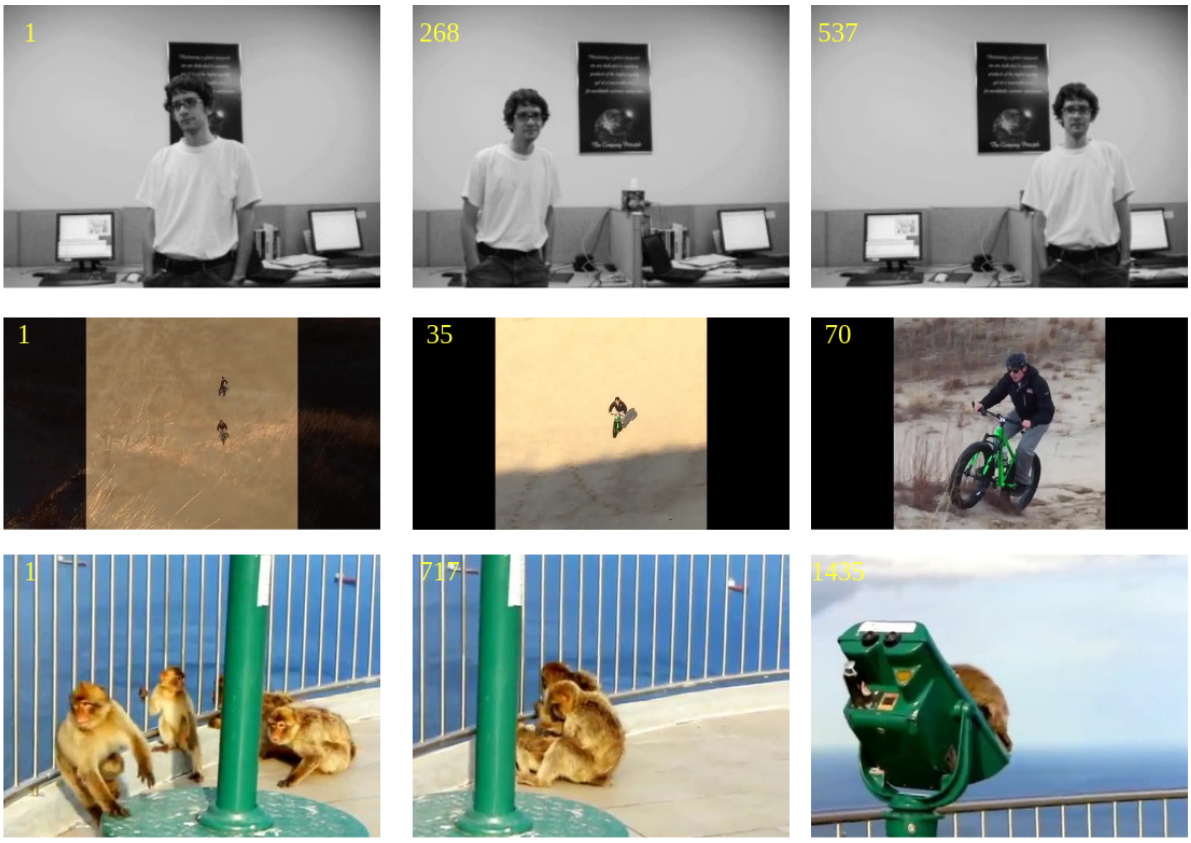


Figure 4.9: Qualitative examples of OTB2015, GOT-10k, LaSOT from top to bottom, respectively.

Table 4.3: A AUC performance table on OTB2015 using precision AUC score (P) and success AUC score (S) to compare TLF-FOKF-related methods and their baseline trackers when the measurement uncertainties are similar. For simplicity, my methods, TLF-FOKF-S, TLF-FOKF-R, and TLF-FOKF-M are namely TFS, TFR, and TFM. The number after $\pm$ shows the std according to the values of each sequence. Red, blue, and green are used to rank the top three performances from high to low.

Method (Block1)	P%	S%	Method	P%	S%
TFS (my)	93.3 ± 14.3	72.7 ± 12.3	SiamRPN++_ResNet	89.1 ± 19.5	68.5 ± 15.2
TFR (my)	93.2 ± 14.5	72.7 ± 12.3	SiamFC++_AlexNet	87.3 ± 21.3	67.6 ± 17.4
TFM (my)	92.7 ± 15.8	72.4 ± 13.1	ECO_HC	84.8 ± 25.4	63.9 ± 20.4
STMTrack	91.7 ± 18.3	71.3 ± 13.6	SiamRPN	83.8 ± 22.7	63.0 ± 16.6

Table 4.4: A performance table on the GOT-10k test set using the average overlap (AO) and success rate (SR) for a comparison between TLF-FOKF-related methods and their baseline trackers. The left and right blocks represent two different tracker ensembles. Because the GOT-10k online evaluation server does not provide the std, they are omitted in this table.

Method (Block1)	AO%	SR%	Method (Block2)	AO%	SR%
TFS (my)	48.6	57.7	**TFS (my)**	63.4	74.9
TFR (my)	48.6	57.7	**TFR (my)**	63.4	74.9
TFM (my)	48.5	57.4	**TFM (my)**	62.4	74.0
SiamRPN++_ResNet	46.4	54.7	STMTrack	62.3	73.8
SiamRPN++_Mobilev2	46.5	54.7	SiamFC++_GoogleNet	59.4	70.4
SiamRPN	45.9	55.6			

Table 4.5: A AUC performance table on LaSOT test set using precision AUC score and success AUC score to compare TLF-FOKF-related methods and their baseline trackers. The left and right blocks represent two different tracker ensembles.

Method (Block1)	P%	S%	Method (Block2)	P%	S%
TFS (my)	50.3 ± 30.8	50.6 ± 24.2	**TFS (my)**	62.3 ± 29.3	62.4 ± 21.8
TFR (my)	50.4 ± 30.5	50.7 ± 23.8	**TFR (my)**	62.3 ± 29.3	62.4 ± 21.8
TFM (my)	49.5 ± 31.0	49.7 ± 24.2	**TFM (my)**	62.3 ± 29.3	62.4 ± 21.8
SiamRPN	44.2 ± 30.6	45.5 ± 24.8	STMTrack	62.4 ± 31.4	60.5 ± 25.7
SiamRPN++_Mobilev2	45.2 ± 31.2	45.1 ± 24.9	STARK	57.8 ± 29.0	58.6 ± 20.9
SiamRPN++_AlexNet	43.6 ± 30.4	44.8 ± 24.1			

measurement uncertainties on the success of them are similar.

An interesting observation from the experiments is that TFS and TFR perform similarly, while TFM often shows worse performance. Regarding the former, this is due to the assumption that all baseline trackers have similar measurement uncertainties, leading to no significant differences between TFS and TFR. Concerning the latter, there are three reasons: Firstly, the decrease in the number of KF trackers s_i results in a lack of comprehensive estimation. Secondly, the variation in the Tracker Ensemble leads to normal experimental results fluctuating. Finally, the estimation of the real model is not accurate enough, resulting in unpredictable experimental outcomes. Nonetheless, these limitations do not impede the effectiveness of TFM in improving tracking performance, especially considering its faster speed of execution. Fig.4.10 presents some qualitative examples of the performance of TLF-FOKF-related (my) in comparison with the baseline trackers.

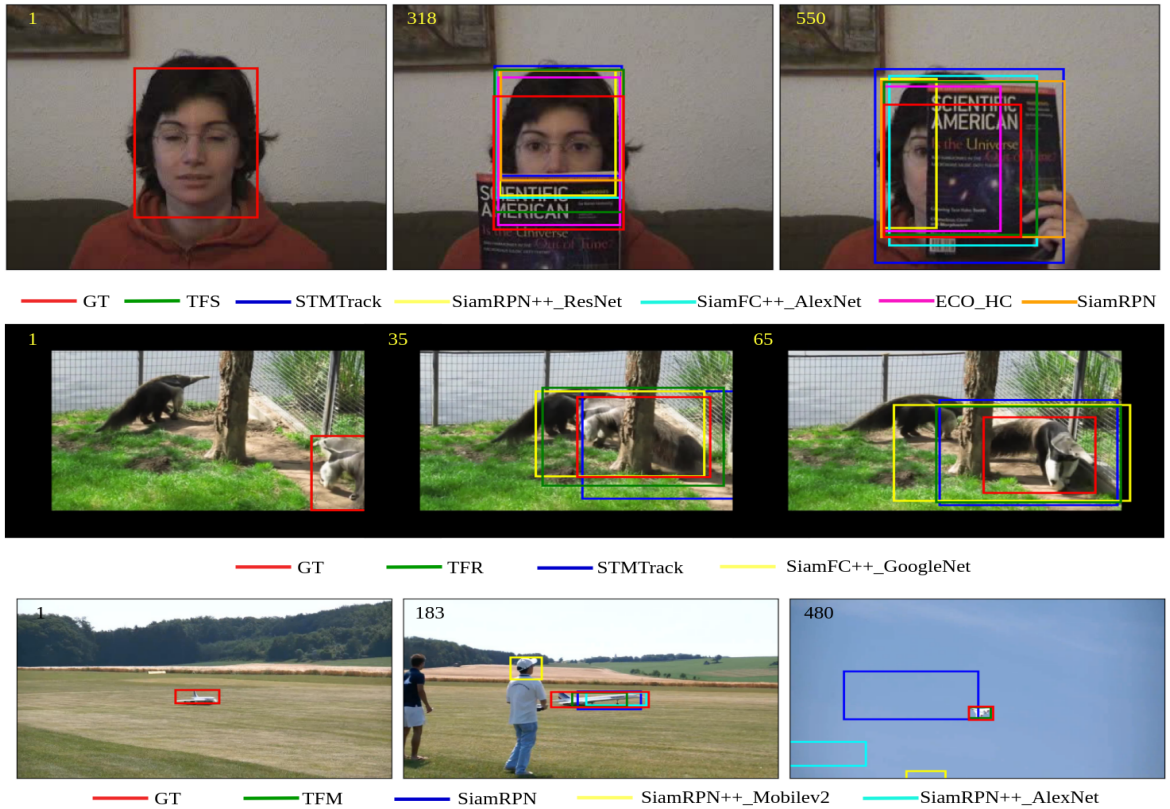


Figure 4.10: Qualitative examples of the tracking ability achieved by the proposed TFS, TFR, and TFM in comparison with the corresponding baseline trackers. The first column of images presents the first frame of each sequence. In the top-left corner of each frame, the frame elapsed since the beginning of the sequence is reported. From top to bottom, the sequences come from the OTB2015, GOT-10k validation set, and LaSOT testing set, respectively. Overall, the solution improves the accuracy and robustness of target tracking by combining the capabilities of the underlying baseline trackers. Additionally, the feedback design of the solution helps the weaker baseline tracker to avoid tracking failure, as indicated in the bottom subplot.

4.2.4. Experiment 2: Tracker Diversity and Acceptability for Feedback

Because I assume R_t^i is the same for each baseline tracker, this experiment only considers the baseline tracker with similar measurement uncertainties as the Tracker Ensemble in order to guarantee the normal performance of TFS, TFR, and TFM. Moreover, I consider the situation that the baseline tracker number is two to decrease the effect of other unknown factors.

To better demonstrate the utilization of tracker diversity, I propose two evaluation metrics, the utilization of the upper bound based on the upper baseline tracker (UU) and the utilization of the upper bound based on the mean baseline tracker (UM). Actually, as for evaluating a TLF method, these two evaluation metrics are more important than simply evaluating the improvement compared with the baseline tracker, because they indicate the utilization ability of a TLF method for the strength of the baseline tracker. The UU metric has practical significance for engineering as it indicates the scale of performance improvement compared with the best baseline tracker, while UM has scientific value for analyzing TLF methods because it considers the average performance of baseline trackers. The definitions are shown as Eqs. 4.9 and 4.10.

$$UU = \frac{TF - UB}{UP - UB} \tag{4.9}$$

$$UM = \frac{TF - MB}{UP - MB} \tag{4.10}$$

where TF is the performance of TLF-FOKF-related methods, and UP is the performance of the Upper Bound which has been described in Section 3.2. UB is the performance of the best baseline tracker in the tracker ensemble, while MB is the mean performance of the baseline trackers in the tracker ensemble.

Table 4.6: A AUC performance table on OTB2015 using success AUC score, UU, UM to compare the performance and the utilization of tracker diversity of TLF-FOKF-related methods and their baseline trackers when the measurement uncertainties are similar. Each block contains one kind of tracker ensemble and the corresponding TFS, TFR, and TFM. Each TLF-FOKF-related method is given corresponding UU and UM in the form of xx/xx in the third column in each block. The first baseline tracker is given the corresponding UP of its tracker ensemble in the third column. For simplicity, some of the baseline tracker names are abbreviated and only the top three performances are highlighted in red, blue, and green from high to low for each evaluation metric and block.

Method	S%	UU/UM%	Method	S%	UU/UM%
TFS	72.4 ± 12.9	27.5/46.3	**TFS**	72.5 ± 12.8	32.4/55.0
TFR	72.4 ± 12.9	27.5/46.3	**TFR**	72.5 ± 12.8	32.4/55.0
TFM	71.8 ± 14.0	12.5/35.2	**TFM**	72.3 ± 12.7	27.0/51.4
STMTrack	71.3 ± 13.6	75.3 ± 11.2	STMTrack	71.3 ± 13.6	75.0 ± 11.1
SiamRPN++_Res	68.5 ± 15.2	Block1	SiamFC++_Alex	67.6 ± 17.4	Block2
TFS	69.2 ± 16.0	13.5/20.4	**TFS**	69.2 ± 17.2	33.3/36.0
TFR	69.2 ± 16.0	13.5/20.4	**TFR**	69.1 ± 17.2	31.2/34.0
TFM	68.8 ± 16.7	5.8/13.3	**TFM**	69.0 ± 16.3	29.2/32.0
SiamRPN++_Res	68.5 ± 15.2	73.7 ± 12.5	SiamFC++_Alex	67.6 ± 17.4	72.4 ± 14.6
SiamFC++_Alex	67.6 ± 17.4	Block3	SiamFC++_Google	67.2 ± 17.2	Block4
TFS	67.7 ± 14.9	43.1/45.9	**TFS**	69.4 ± 14.7	61.1/63.0
TFR	67.7 ± 14.9	43.1/45.9	**TFR**	68.8 ± 15.6	54.4/56.6
TFM	67.2 ± 16.1	34.5/37.7	**TFM**	68.9 ± 15.1	55.6/57.7
SiamRPN++_Mobile	65.2 ± 18.1	71.0 ± 14.9	ECO_HC	63.9 ± 20.4	72.9 ± 13.0
SiamRPN++_Alex	64.6 ± 16.9	Block5	SiamRPN	63.0 ± 16.6	Block6

The group design of the above experiments is based on the ranking of the mean performance of baseline trackers from high to low, as shown in Table 4.6. These experimental results not only again demonstrate the performance improved by TLF-FOKF-related but also show the positive correlation between the fusion performance and the tracker diversity proposed by Section 3.2. For example, the first row of Table 4.6 TLF-FOKF-related have a better performance when their upper bounds (UP) also have the highest success AUC score around 75.3%, compared with other tracker ensembles on OTB2015. Moreover, a strong evidence example is the tracker ensemble with ECO_HC and SiamRPN. It has a

Table 4.7: A performance table on GOT-10k validation set using average overlap (AO), UU, UM to compare the performance and the utilization of tracker diversity of TLF-FOKF-related methods and their baseline trackers when the measurement uncertainties are similar. Here, the validation set is used for analysis convenience of std. And it only indicates the best performance in red. The rest is the same as Table 4.6.

Method	AO%	$UU/UM\%$	Method	AO%	$UU/UM\%$
TFS	75.4 ± 17.7	20.7/27.6	**TFS**	64.6 ± 21.8	22.9/28.2
TFR	75.4 ± 17.7	20.7/27.6	**TFR**	64.6 ± 21.8	22.9/28.2
TFM	74.9 ± 18.1	12.1/19.7	**TFM**	64.3 ± 21.9	16.7/22.3
SiamFC++_Google	74.2 ± 19.8	80.0 ± 14.8	SiamRPN++_Mobile	63.5 ± 23.1	68.3 ± 20.8
STARK	73.1 ± 18.2	Block1	SiamRPN++_Alex	62.8 ± 20.9	Block2

Table 4.8: A performance table on LaSOT testing set using success AUC score, UU, UM to compare the performance and the utilization of tracker diversity of TLF-FOKF-related methods and their baseline trackers when the measurement uncertainties are similar. The rest is the same as Table 4.6.

Method	S%	$UU/UM\%$	Method	S%	$UU/UM\%$
TFS	62.4 ± 21.8	20.7/28.1	**TFS**	48.0 ± 24.0	33.7/34.9
TFR	62.4 ± 21.8	20.7/28.1	**TFR**	48.4 ± 23.9	38.4/39.4
TFM	62.4 ± 21.8	20.7/28.1	**TFM**	48.7 ± 23.7	41.9/42.9
STMTrack	60.5 ± 25.7	69.7 ± 19.0	SiamRPN++_Mobile	45.1 ± 24.9	53.7 ± 23.6
STARK	58.6 ± 20.9	Block1	SiamRPN++_Alex	44.8 ± 24.1	Block2

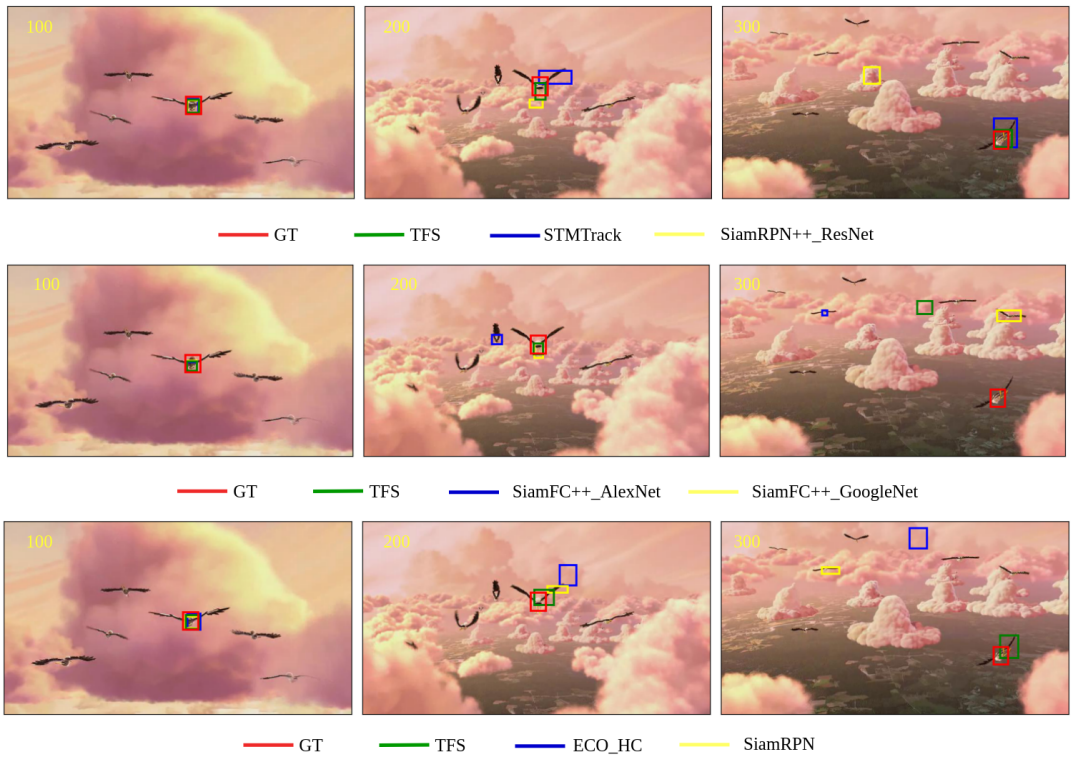


Figure 4.11: Qualitative examples for experiment 2 achieved by the proposed TFS in comparison on OTB2015 with the corresponding baseline trackers. From top to bottom, there are three different tracker ensembles and three different tracker diversity listed in Table 4.6. Overall, the top sub-figure has the highest tracker diversity, and hence TFS performs best, while the other two tracker ensembles have similar tracker diversity, and their TFS have similar tracking performance. Note that although tracker diversity is crucial for fusion performance, there are many impact factors, such as the acceptability of the baseline tracker for feedback and the TLF method itself.

much lower mean baseline tracker success AUC score (around 63.5%) but its TFS has an outstanding fusion performance (69.4%). Another persuasive metric is to calculate the Pearson correlation coefficient according to Tables 4.6-4.8, where the Pearson correlation coefficient ranges from -1 to 1, with a value of -1 indicating a perfect negative correlation, a value of 0 indicating no linear correlation, and a value of 1 indicating a perfect positive correlation. Table 4.9 demonstrates the strong positive correlation between the tracker diversity and the TLF-FOKF-related methods with the Pearson correlation coefficients of about 0.8.

Table 4.9: The Pearson correlation coefficient about TLF-FOKF-related and the upper bound according to Tables 4.6-4.8, 4.11-4.13 and Figs.4.12-4.14 with the total 24 data where the estimated measurement uncertainties are the same.

Method	**TFS**	**TFR**	**TFM**
Coefficient	0.8198	0.8203	0.8030

It is worth noting that the UU values of TLF-FOKF-related methods are primarily concentrated in the interval of [20,40], while the UM values are concentrated in the interval of [30,50]. These two metrics provide a clear indication of the scale of the utilization of the upper bound of TLF-FOKF-related methods.

There is evidence to support the acceptability of baseline trackers for feedback. Comparing the success AUC scores of Block1 and Block2 in Table 4.6, we can clearly observe that the success AUC scores of TLF-FOKF-related containing the baseline tracker SiamRPN++_ResNet always performs worse, although their upper bound and the mean baseline tracker performance are higher. It can infer that the acceptability of SiamRPN++_ResNet for feedback is worse than others because the other variables are similar. Some qualitative examples of the performance of TFS in comparison with different baseline trackers on OTB2015 are shown in Fig.4.11.

4.2.5. Experiment 3: Baseline Trackers' Number

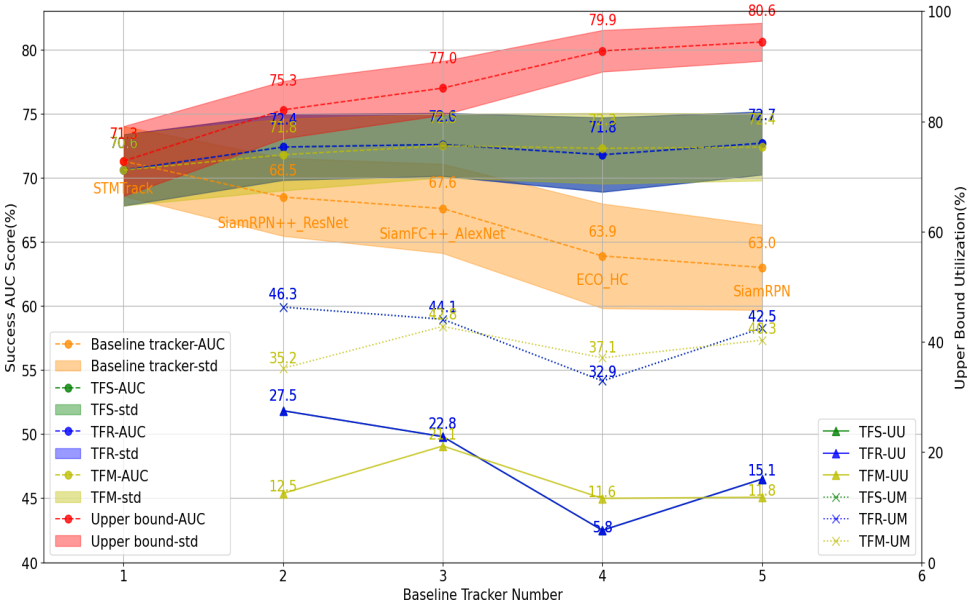


Figure 4.12: The influence of baseline trackers' number with similar measurement uncertainties on OTB2015 for performance and utilization of tracker diversity, where the baseline tracker is added one by one, including STMTrack, SiamRPN++_ResNet, SiamFC++_AlexNet, ECO_HC, SiamRPN. The top side corresponds to the success AUC score, while the bottom side corresponds to the utilization of tracker diversity by UU and UM. The std is calculated based on each sequence and shown on the graph after being scaled down by a factor of 5 for aesthetics.

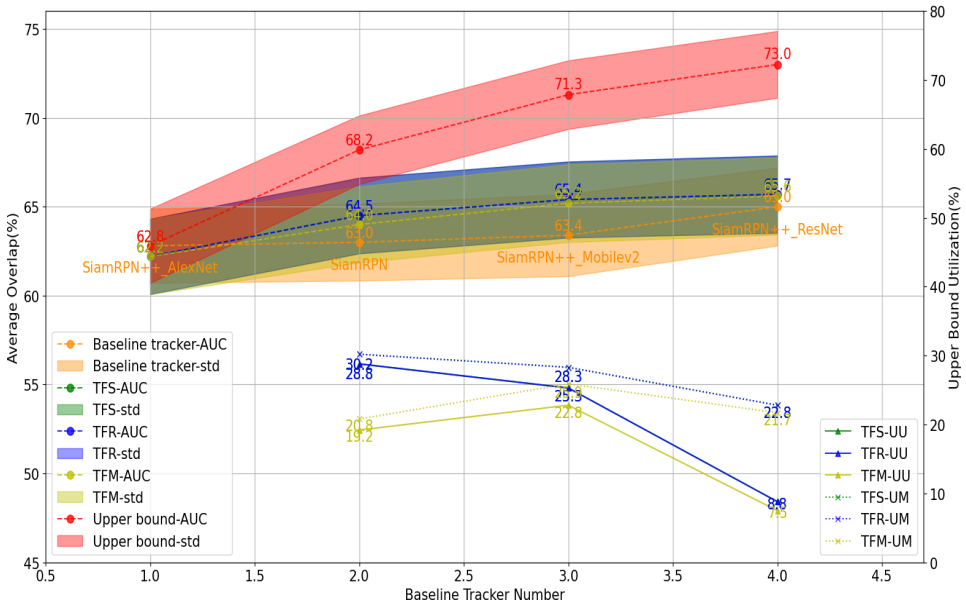


Figure 4.13: The influence of baseline trackers' number with similar measurement uncertainties on GOT-10k validation set for performance and utilization of tracker diversity. The std is calculated based on each sequence and shown on the graph after being scaled down by a factor of 10 for aesthetics.

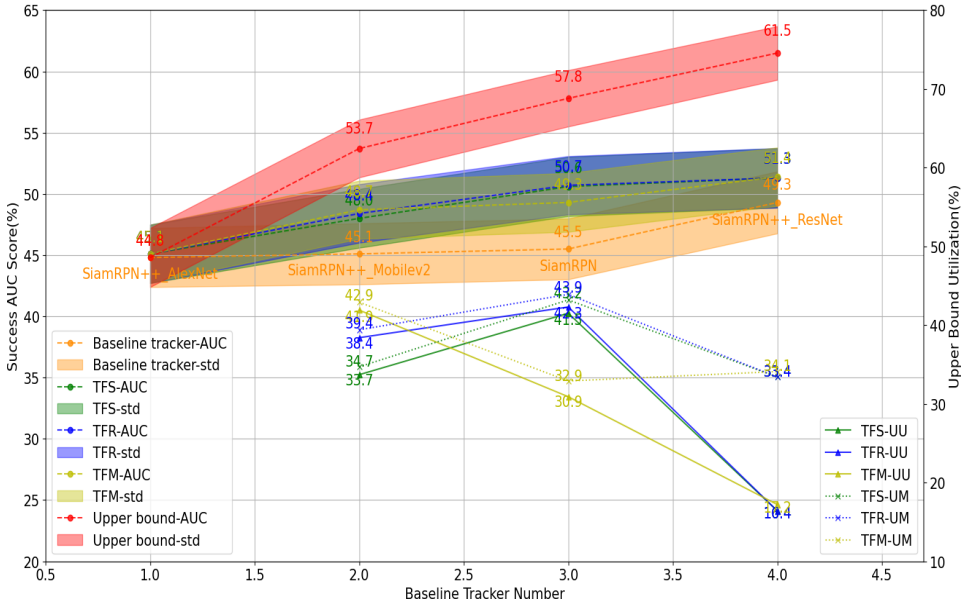


Figure 4.14: The influence of baseline trackers' number with similar measurement uncertainties on LaSOT testing set for performance and utilization of tracker diversity. The std is calculated based on each sequence and shown on the graph after being scaled down by a factor of 10 for aesthetics.

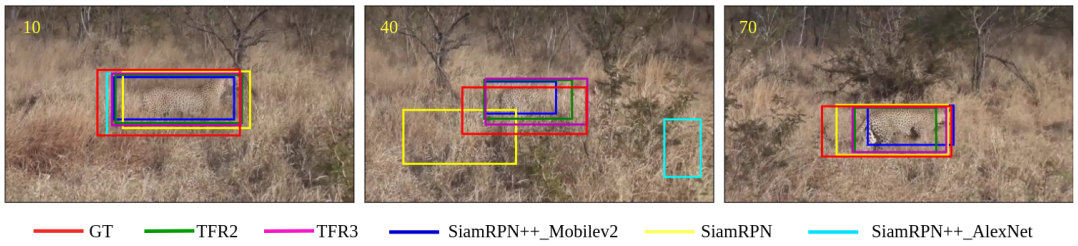


Figure 4.15: Qualitative examples for experiment 3 achieved by the proposed TFR in comparison on the GOT-10k validation set with two different number baseline trackers. Here, TFR2 uses the tracker ensemble: SiamRPN++_Mobilev2 and SiamRPN, while TFR3 adds one more baseline tracker (SiamRPN++_AlexNet) based on the tracker ensemble of TFR2, which is a qualitative example of Fig.4.13. Comparing TFR2 and TFR3, better performance is achieved by TFR3, especially in frame 40.

A key advantage of the TLF method is its ability to enhance performance with an increasing number of baseline trackers. However, the efficacy of this advantage is subject to various challenging factors, including the TLF method's own prerequisites, negative effects as proposed in Section 3.2, and others. Hence, this section primarily aims to investigate the influence of the number of baseline trackers on TLF-FOKF-related methods, under the constraint of maintaining the TLF-FOKF-related method's feasibility. Experimental results demonstrating the findings are presented in Figs. 4.12-4.14.

Figs. 4.12-4.14 all clearly show that the fusion performance of TLF-FOKF-related is improved with the addition of the number of baseline trackers. As for the accuracy performance, Fig.4.12 adds the baseline tracker from high to low based on their performance. It is obvious that the success AUC scores increase while the std decrease with the growth of the baseline tracker's number. On the contrary, the AUC and std of the baseline trackers are opposite. The AUC of TLF-FOKF-related is also increased with the number of baseline trackers, except after adding ECO_HC. This sudden drop is because the measurement uncertainty of ECO_HC is vastly dissimilar to that of other baseline trackers, causing some extent of performance degradation. Similar situations are also observable in Fig.4.13 and 4.14 after adding SiamRPN++_ResNet. However, the performance improvement of TLF-FOKF-related is satisfactory when the measurement uncertainties are close enough because all baseline trackers have the same R_t^i. Analysis of Fig.4.12-4.14 indicates that TFS and TFR typically have comparable accuracy performance. TFM performs somewhat more poorly when the number of baseline trackers is small but performs better when the number of baseline trackers is sufficiently large. Moreover, the negative effect of a limited number of baseline trackers is also evident based on these analyses and figures. Using more baseline trackers brings more tracker diversity, but also brings more negative effects, which also need to be considered.

Regarding the utilization of the upper bound, UU mostly falls in the range $[10, 40]$, while UM is concentrated in the interval $[25, 45]$, making it more stable. The stability of UM confirms that TLF-FOKF-related can harness tracker diversity to improve tracking performance consistently. In contrast, the fluctuation in UU is primarily due to the imprecise estimation of R_t^i. Fig.4.15 shows some qualitative examples of TFR's performance concerning different numbers of baseline trackers on the GOT-10k validation set.

4.2.6. Experiment 4: Running Speed

Table 4.10: Comparison of the running speed of TLF methods, where the top two methods are highlighted in red and blue.

method	Xie et al.	Dunnhofer et al.	W. Zhang et al.	Dunnhofer et al.	**TFS/TFR**	**TFM**
Year	2019	2020	2020	2022	2023	2023
FPS	60~500	30	200	20~90	160~630	820~1100

The running speed of a TLF method is a crucial evaluation metric since the TLF method's running speed plus the baseline trackers are inherently slower than that of the baseline trackers. Thus, knowing the operational speed of TLF-FOKF-related is crucial in determining whether it significantly affects tracking speed in practical applications. Fig.4.16 indicates that TFS and TFR's running speed exhibits a quadratic pattern, allowing the fastest operation to reach 630 FPS with two baseline trackers, while

the slowest reaches over 150 FPS with seven baseline trackers. On the other hand, TFM's running speed demonstrates a linear pattern, with the slowest operation recorded at over 800 FPS with seven baseline trackers. Such running speeds enable TLF-FOKF-related to achieve effective real-time fusion. The total speed equals the sum of the baseline trackers' speed (either sequential or parallel) and the speed of TLF-FOKF-related.

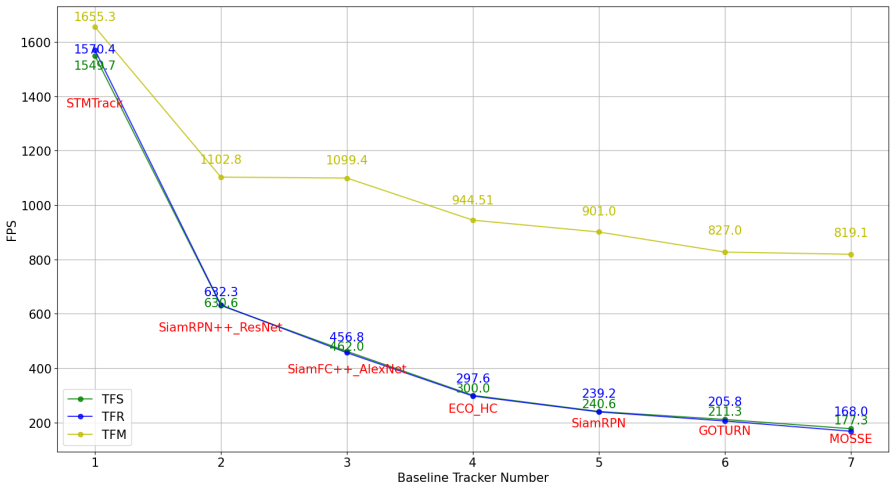


Figure 4.16: The running speed of TFS, TFR, and TFM, which is tested on the sequence Basketball of OTB2015. The reason for only testing on one sequence is the running speed of TLF-FOKF-related mainly related to the number of trackers while independent of the sequence. Here the selection of baseline trackers is only for testing speed without considering their capabilities.

In addition, a comparison of the state-of-the-art TLF method running speeds can be made based on their respective literature, as depicted in Table 4.10. Notably, TFS and TFR exhibit superior running speeds, with TFR outpacing all other methods.

4.2.7. Experiment 5: Dissimilar Measurement Uncertainties and Negative Effects

The preceding experiments mostly considered baseline trackers with comparable measurement uncertainties. However, dissimilar measurement uncertainties are more prevalent in practical engineering applications. Thus, it is pivotal to analyze the performance of TLF-FOKF-related concerning dissimilar measurement uncertainties. Such disparate uncertainties are occasionally associated with negative effects, such as poor tracking outcomes, which become particularly evident in comparisons between robust and weak baseline trackers. Hence, the experiments presented herein will assess such negative effects.

The reason why similar measurement uncertainties were employed for all baseline trackers in the preceding experiments is that accurately gauging the measurement uncertainties for all baseline trackers and frames remains challenging. Nonetheless, this does not imply that TLF-FOKF-related methods are inadequate for handling dissimilar measurement uncertainties. To demonstrate this, succeeding experiments utilize rough hand-crafted measurement uncertainties, as depicted in Tables 4.11-4.13.

Tables 4.11-4.13 demonstrate that TLF-FOKF-related methods exhibit considerably better performance following the utilization of more suitable measurement uncertainties. This strongly suggests that TLF-FOKF-related methods possess the potential to perform optimally under varying measurement uncertainties provided that their estimation is accurate enough. When the number of baseline trackers is relatively large and rough hand-crafted measurement uncertainties are used, TFR-Score frequently exhibits the best performance, while TFS-Score fares the worst, with TFM-Score ranking in between. The resulting data support the Table 3.2 proposition.

In Section 3.2, it is highlighted that STARK and MOSSE inevitably produce poor tracking results, as evidenced in Tables 4.11 and 4.13, which demonstrate that TLF-FOKF-based trackers perform poorly when STARK and MOSSE are used as the baseline trackers. The negative impact of these poor track-

Table 4.11: A AUC performance table on OTB2015 using success AUC score, UU, UM to compare the performance and the utilization of tracker diversity after using different R_t^i when the measurement uncertainties are dissimilar. Each horizontal block represents one kind of tracker ensemble, while the TLF-FOKF-related suffix with 'Score' in each vertical second column block represents that the estimated R_t^i is different. For each vertical second block, it uses $R_t^1 = 0.1R_t$ and $R_t^2 = R_t$. TLF-FOKF-related are compared with TLF-FOKF-related-Score, where better performance is indicated in red.

Method	AUC%	$UU/UM\%$	Method	AUC%	$UU/UM\%$
TFS	65.3 ± 18.3	$-157.9/-24.8$	**TFS-Score**	70.0 ± 14.6	$-34.2/35.0$
TFR	65.2 ± 18.3	$-160.5/-26.1$	**TFR-Score**	70.0 ± 14.6	$-34.2/35.0$
TFM	64.9 ± 18.7	$-168.4/-29.9$	**TFM-Score**	70.2 ± 14.0	$-28.9/37.6$
STMTrack[1]	71.3 ± 13.6	75.1 ± 10.2	STARK[2]	63.2 ± 18.4	Block1
TFS	71.0 ± 13.1	$-17.6/89.0$	**TFS-Score**	71.2 ± 12.8	$-5.9/90.1$
TFR	71.0 ± 13.1	$-17.6/89.0$	**TFR-Score**	71.2 ± 12.8	$-5.9/90.1$
TFM	70.6 ± 14.2	$-41.2/86.7$	**TFM-Score**	70.6 ± 14.2	$-41.2/86.7$
STMTrack[1]	71.3 ± 13.6	73.0 ± 11.9	GOTURN[2]	38.5 ± 22.6	Block2

Table 4.12: A performance table on GOT-10k validation set using average overlap (AO), UU, UM to compare the performance and the utilization of tracker diversity after using different R_t^i when the measurement uncertainties are dissimilar. For the horizontal first vertical second block, it uses $R_t^1 = 0.1R_t$ and $R_t^2 = R_t$. For the horizontal second vertical second block, it uses $R_t^1 = 0.1R_t$, $R_t^2 = 0.7R_t$, $R_t^3 = 0.9R_t$, $R_t^4 = R_t$.

Method	AO%	$UU/UM\%$	Method	AO%	$UU/UM\%$
TFS	65.2 ± 20.3	$-352.6/5.2$	**TFS-Score**	77.5 ± 16.9	$-28.9/73.0$
TFR	65.2 ± 20.3	$-352.6/5.2$	**TFR-Score**	77.5 ± 16.9	$-28.9/73.0$
TFM	63.8 ± 21.5	$-389.5/-2.5$	**TFM-Score**	76.4 ± 17.4	$-57.9/66.9$
STMTrack[1]	78.6 ± 17.5	82.4 ± 13.7	ECO_HC[2]	49.9 ± 25.0	Block1
TFS	76.5 ± 17.1	$-67.7/54.8$	**TFS-Score**	77.4 ± 17.4	$-38.7/62.6$
TFR	76.5 ± 17.1	$-67.7/54.8$	**TFR-Score**	78.6 ± 17.1	$0.0/73.0$
TFM	76.2 ± 17.1	$-77.4/52.2$	**TFM-Score**	78.4 ± 16.8	$-6.5/71.3$
STMTrack[1]	78.6 ± 17.5	81.7 ± 15.7	SiamFC++_Google[2]	74.2 ± 19.8	–
SiamRPN++_Res[3]	65.0 ± 21.9	–	SiamRPN[4]	63.0 ± 21.6	Block2

Table 4.13: A performance table on LaSOT testing set using success AUC score, UU, UM to compare the performance and the utilization of tracker diversity after using different R_t^i when the measurement uncertainties are dissimilar. For the horizontal first vertical second block, it uses $R_t^1 = 0.5R_t$ and $R_t^2 = R_t$. For the horizontal second vertical second block, it uses $R_t^1 = 0.1R_t$, $R_t^2 = 0.3R_t$, $R_t^3 = 0.7R_t$, $R_t^4 = 0.9R_t$, $R_t^5 = R_t$, $R_t^6 = 1.2R_t$.

Method	AUC%	$UU/UM\%$	Method	AUC%	$UU/UM\%$
TFS	60.2 ± 25.2	$-6.7/31.4$	**TFS-Score**	60.6 ± 25.0	$2.2/37.1$
TFR	60.0 ± 25.2	$-11.1/28.6$	**TFR-Score**	60.6 ± 25.0	$2.2/37.1$
TFM	60.1 ± 24.9	$-8.9/30.0$	**TFM-Score**	60.4 ± 24.9	$-2.2/34.3$
STMTrack[1]	60.5 ± 25.7	65.0 ± 23.6	SiamFC++_Google[2]	55.5 ± 24.8	Block1
TFS	44.1 ± 21.3	$-205.0/21.8$	**TFS-Score**	49.2 ± 21.6	$-141.2/38.1$
TFR	44.3 ± 20.8	$-202.5/22.4$	**TFR-Score**	56.6 ± 23.1	$-48.7/61.8$
TFM	42.7 ± 21.3	$-222.5/17.3$	**TFM-Score**	54.6 ± 23.1	$-73.7/55.4$
STMTrack[1]	60.5 ± 25.7	68.5 ± 21.8	SiamFC++_Google[2]	55.5 ± 24.8	–
SiamRPN++_Mobile[3]	45.1 ± 24.9	–	ECO_HC[4]	29.4 ± 22.5	–
GOTURN[5]	21.3 ± 20.3	–	MOSSE[6]	12.1 ± 15.1	Block2

ing results severely affects the performance of TLF-FOKF-based trackers. In addition, the performance of TLF-FOKF-based trackers in Block2 of Table 4.12 is also weak, despite the fact that the four baseline trackers used in this block do not have any notable drawbacks. This indicates that the occasional poor tracking results have a detrimental effect on the performance of TLF-FOKF-based trackers. Notably, the second block in Table 4.11 registers the highest UM among the three tables. This can be attributed to the fact that the GOTURN tracking object is entirely based on the previous frame's estimation and is, therefore, relatively more receptive to feedback.

4.2.8. Experiment 6: Process Uncertainty

In addition to measurement uncertainty R_t^i, process uncertainty Q_t is also an important hyperparameter for TLF-FOKF-related. The process uncertainty reflects the accuracy of the object motion model A_t (the state transition matrix). Therefore, this section aims to evaluate the influence of process uncertainty on the tracking performance as demonstrated in Tables 4.14 to 4.16.

Table 4.14: A AUC performance table on OTB2015 using precision AUC score (P) and success AUC score (S) to compare the performance and the utilization of tracker diversity after using different Q_t when the measurement uncertainties are similar. The TLF-FOKF-related suffix with 'Q' in the fourth column represents that the estimated Q_t is different. Here, it uses $Q_t^{new} = 0.01Q_t^{old}$. TLF-FOKF-related are compared with TLF-FOKF-related-Q, where better performance is indicated in red.

Method (Block1)	P%	S%	Method	P%	S%
TFS	92.4 ± 16.9	71.8 ± 14.5	**TFS-Q**	92.0 ± 18.0	71.2 ± 15.0
TFR	92.4 ± 16.9	71.8 ± 14.5	**TFR-Q**	91.2 ± 18.9	70.8 ± 15.3
TFM	92.9 ± 15.7	72.3 ± 13.8	**TFM-Q**	91.5 ± 18.1	71.0 ± 14.9
STMTrack	91.7 ± 18.3	71.3 ± 13.6	SiamRPN++_Res	89.1 ± 19.5	68.5 ± 15.2
SiamFC++_Alex	87.3 ± 21.3	67.6 ± 12.0	ECO_HC	84.8 ± 25.4	63.9 ± 20.4

Table 4.15: A performance table on GOT-10k validation set using the average overlap (AO) and success rate (SR) to compare the performance and the utilization of tracker diversity after using different Q_t when the measurement uncertainties are similar. Here, it uses $Q_t^{new} = 0.01Q_t^{old}$.

Method (Block1)	AO%	SR%	Method	AO%	SR%
TFS	65.4 ± 21.4	76.7 ± 30.7	TFS-Q	64.1 ± 21.8	75.5 ± 31.0
TFR	65.4 ± 21.4	76.7 ± 30.7	TFR-Q	64.1 ± 21.8	75.5 ± 31.0
TFM	65.2 ± 21.9	76.4 ± 31.1	TFM-Q	63.7 ± 22.1	75.0 ± 31.2
SiamRPN++_Mobile	63.4 ± 23.1	74.5 ± 31.9	SiamRPN	63.0 ± 21.6	75.0 ± 31.2
SiamRPN++_Alex	62.8 ± 20.9	74.3 ± 30.5			

Table 4.16: A AUC performance table on LaSOT testing set using precision AUC score (P) and success AUC score (S) to compare the performance and the utilization of tracker diversity after using different Q_t when the measurement uncertainties are similar. Here, it uses $Q_t^{new} = 0.01Q_t^{old}$.

Method (Block1)	P%	S%	Method	P%	S%
TFS	50.3 ± 30.8	50.6 ± 24.2	TFS-Q	49.8 ± 20.6	50.2 ± 24.0
TFR	50.4 ± 30.5	50.7 ± 23.8	TFR-Q	49.8 ± 30.6	50.2 ± 24.0
TFM	49.5 ± 31.0	49.7 ± 24.2	TFM-Q	48.3 ± 30.4	48.6 ± 24.0
SiamRPN	44.2 ± 30.6	45.5 ± 24.8	SiamRPN++_Mobile	45.2 ± 31.2	45.1 ± 24.9
SiamRPN++_Alex	43.6 ± 30.4	44.8 ± 24.1			

From the results presented in the aforementioned tables, it is evident that for TLF-FOKF-related trackers which utilize the CV model for object motion modeling, a lower process uncertainty leads to over-belief in the CV model, which in turn mildly reduces the tracking performance. This outcome suggests that the CV model is not capable of adequately capturing the dynamics of real object movements. Furthermore, it emphasizes that the hyperparameter Q_t plays a vital role in the tracking process and cannot be ignored, despite its influence being less significant than R_t^i.

4.2.9. Comparison with the state-of-the-art TLF Methods

Comparing TLF-FOKF-related trackers to the state-of-the-art TLF methods is more challenging than comparing with the baseline tracker methods. A fair comparison requires the use of the same tracker ensemble; however, many TLF methods have unique designs for their tracker ensembles, making it nearly impossible to achieve fairness in this context. Additionally, most TLF methods do not have openly accessible code repositories, further complicating the process of reproducing their results. Hence, in this section, I present a comparison between TLF-FOKF-related and other TLF methods based on a thorough analysis of their respective papers to ensure a fair comparison as far as possible.

The best performance (S:65.0%, P:83.2%) of the work of Bailer et al. is tested on OTB2013 (Wu et al., 2013) which is a subset of OTB2015. They used 29 baseline trackers from (Wu et al., 2013) with the success AUC score interval [21.0%, 50.5%]. The improvement of success AUC score is up to 14.5%. As for TFS, it uses 3 baseline trackers of my work with success AUC scores about 65.0% and reaches the fusion performance up to 70.9%, where the improvement is up to 5.9%. Since the improvement in tracking performance becomes progressively more challenging as the performance approaches the upper bound of a dataset, and given the remarkable difference in the number of baseline trackers, TLF-FOKF-related trackers continue to be a competitive alternative to the work of Bailer et al. (2014), despite the variations in the baseline trackers and tracking performance measures.

Similarly, the work of Xie et al. (2019) can reach the success AUC score 69.0% on OTB2015 by using 10 baseline trackers with the success AUC score interval [53.4%, 64.8%]. TFS using 2 baseline trackers with success AUC scores about 63.5% can reach 69.4%. Both of the aforementioned tracker ensembles utilize ECO_HC as a baseline tracker. In such a scenario, it is evident that TLF-FOKF-related trackers perform considerably better, particularly when the number of baseline trackers is small.

Moreover, Dunnhofer et al. (2020) used the tracker ensemble (SiamRPN + ECO_HC) and obtained the success AUC score 67.0%, while TFS uses the same tracker ensemble and reaches 69.4%, which is a clear fair comparison with the better performance. In addition, W. Zhang et al. (2020) used SiamRPN++_ResNet (S:49.6%) and ATOM (Danelljan et al., 2019) (S:51.5%) as the baseline trackers and achieved the fusion performance of success AUC score 51.6% on LaSOT, while TFS uses SiamRPN++_ResNet (S:49.3%) and SiamFC++_AlexNet (S:51.4%) as the baseline trackers and achieves 54.2% which is prominently better, although these two tracker ensembles are not the completely same.

The studies mentioned above are not excessively critical of the choice of baseline tracker. But the work of Dunnhofer et al., 2022 (2022) required to use of the complementary trackers, SuperDiMP (Bhat et al., 2019 and Danelljan et al., 2020) (S:63.1%) and STARK-ST50 (S:66.4%), and obtained the outstanding success AUC score 68.5% on LaSOT. Although the baseline trackers in this thesis do not have similar performance as the above two trackers, TFS can use SiamRPN++_ResNet and SiamFC++_AlexNet as the baseline trackers with the fusion performance 54.2% to compare. Based on the extent of performance improvement and difference observed between the two chosen baseline trackers, we can conclude that TLF-FOKF-related trackers have a comparable fusion performance to the approach by Dunnhofer et al. (2022).

Based on the comparison made above, assuming the same measurement uncertainty for the baseline trackers, TLF-FOKF-related trackers in their current form already demonstrate state-of-the-art tracking performance.

4.2.10. Analysis

This section provides a general summary of the experimental results presented above, along with a discussion of the advantages and disadvantages of the TLF-FOKF method. Furthermore, I examine the implications of our findings and their potential impact.

As for the strengths of TLF-FOKF-related, they can be summarized as follows:

1. **Superior Fusion Performance**: The TLF-FOKF method provides remarkable performance improvements compared to baseline trackers, evidenced by UM interval concentrations between 25 and 45. As a TLF method, TLF-FOKF achieves state-of-the-art fusion performance when hyperparameters are accurately selected.

2. **Fast Execution Speed**: The TLF-FOKF method executes extremely fast, with the lowest TFM speed up to 820 FPS, as reported in Section 4.2.6.

3. **Explainable Model**: The TLF-FOKF method is a white-box model that can explain the generation of fusion results with clarity.
4. **Cost-Effective Implementation**: The TLF-FOKF method has low storage and hardware cost while only saving the previous frame states, which can be easily implemented by a few equations.
5. **Insensitive to Number of Baseline Trackers**: Unlike Xie et al. (2019), which requires at least three baseline trackers and performs best with eight or more, the TLF-FOKF method has no requirement for the number of baseline trackers.

The drawbacks of TLF-FOKF-related can be summarized as follows:

1. **Hyperparameters Estimation**: The TLF-FOKF method does not resolve the problem of estimating hyperparameters A_t, Q_t, and R_t^i, attributing to poor baseline tracker quality that can degrade the fusion performance.
2. **Limitations of Linear System and Gaussian Distribution**: The TLF-FOKF model, like the KF, is based on the linear system assumption and Gaussian distribution of process and measurement noise. This assumption limits the model's versatility in cases where the actual model is non-Gaussian or nonlinear.

In addition to TLF-FOKF-related, I have also demonstrated a strong correlation between tracker diversity and TLF fusion performance, as well as the adverse effects of poor baseline trackers. These findings have significant implications and can guide future research on the VOT task and the TLF topic.

5 Conclusion

This section provides an overview of my entire thesis, outlines areas for future research based on current limitations, and offers practical recommendations for the application of TLF-FOKF-related.

5.1. Summary

The Introduction chapter provides an overview of the VOT task and its challenges. The motivation for this thesis is explained, where the potential of TLF methods for the VOT task is identified along with the gap in employing Distributed KF. The chapter concludes by proposing three research questions to address this gap and providing a brief summary of the contributions of the thesis.

The Related Work chapter provides a detailed introduction to the general framework of a VOT tracker and the existing baseline trackers for the VOT task. TLF methods and the key methodology, Distributed KF, are also described for further understanding. The chapter concludes with a summary of observations about the above works and the identification of research gaps.

In the Methodology chapter, I present my work. Firstly, I provide a summary of the current TLF methods and propose a general pipeline for TLF methods, which is aimed at answering the first research question in Section 1.3. This proposal is meant to serve as a guideline for creating TLF methods. I also conduct an in-depth analysis of the impacts of baseline trackers to answer the second research question, discussing the positive effects (tracker diversity) and four negative effects. This analysis has practical implications for engineers and researchers and can aid in explaining various experimental results. Subsequently, I propose FOKF-Standard for the VOT scenario, motivated by our previous work (Zhong and Liu, 2021). This method revises the definition of measurements to answer the third research question. FOKF-Standard is a Distributed Kalman Filter that collaboratively estimates states using local and community measurements. Additionally, I present TLF-FOKF-Standard, which combines FOKF-Standard with Tracker Ensemble. To tackle the issue of FOKF-Standard performing better only under similar measurement uncertainties, I introduce the FOKF-R method. This method further enhances performance by eliminating the weight step of FOKF-Standard. Similarly, I propose TLF-FOKF-R for VOT by selecting the minimal state covariance matrix $P^i_{t|t}$ as the fusion result. To further reduce time complexity and utilize vague prior knowledge, I develop FOKF-M by compressing the number of KF trackers and selecting the optimal baseline tracker as the primary tracker. TLF-FOKF-M is presented in a similar manner.

In the Experiments chapter, I evaluate the performance of FOKF-related and TLF-FOKF-related methods under different experimental conditions. For FOKF-related methods, I conduct experiments using ideal variables that follow a Gaussian distribution, where measurement uncertainties, whether known or unknown, and whether similar or dissimilar, are tested. The results show that FOKF-Standard performs best when measurement uncertainties are similar, while FOKF-R is optimal when uncertainties are dissimilar and known, and FOKF-M outperforms the other two when uncertainties are dissimilar but unknown. For TLF-FOKF-related methods, I test them on three challenging VOT benchmarks, including experiments on comparing them with baseline trackers, evaluating tracker diversity and acceptability of baseline trackers for feedback, analyzing the influence of the number of baseline trackers and running speed, studying dissimilar measurement uncertainties and negative effects, and evaluating the

impact of process uncertainty. In addition, I provide inference-based comparisons with state-of-the-art TLF methods. These experiments clearly demonstrate that TLF-FOKF-related methods have superior fusion performance and support my hypothesis regarding their positive and negative effects. Finally, I conclude by discussing the strengths and limitations of TLF-FOKF-related methods and the potential impact of my findings.

5.2. Future Works

The main limitation of TLF-FOKF-related methods is the difficulty in accurately estimating hyperparameters such as A_t, Q_t, and R_t^i. Addressing this challenge successfully could significantly enhance the fusion performance of TLF-FOKF-related methods, particularly when measurement uncertainties differ considerably or when dealing with negative effects discussed in Section 3.2. Therefore, future work should focus on developing approaches to estimate hyperparameters accurately, which is the first crucial step in advancing this area of research.

5.3. Recommendations

Although TLF-FOKF-related methods are not perfect and have limitations, it is remarkable that they can improve overall tracking performance by fusing multiple baseline trackers. The simplicity, efficiency, interpretability, and low requirements of these methods make them tremendously valuable in practice. For industrial applications, engineers could integrate two or three baseline trackers as trackers to enhance tracking performance, maintain cost control, and ensure robustness to some extent.

Moreover, the success of FOKF in VOT underscores the enormous potential of FOKF in other areas. For instance, rather than relying on baseline trackers, we can employ multiple sensors to improve tracking accuracy. In addition, the applications of FOKF need not be limited to VOT but could be extended to multi-target tracking, object detection, pose estimation, and beyond. Future research could further explore the potential applications of FOKF in various fields.

Bibliography

Bailer, C., Pagani, A., & Stricker, D. (2014). A superior tracking approach: Building a strong tracker through fusion. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*, 170–185.

Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., & Torr, P. H. (2016). Fully-convolutional siamese networks for object tracking. *European conference on computer vision*, 850–865.

Bhat, G., Danelljan, M., Gool, L. V., & Timofte, R. (2019). Learning discriminative model prediction for tracking. *Proceedings of the IEEE/CVF international conference on computer vision*, 6182–6191.

Biresaw, T. A., Cavallaro, A., & Regazzoni, C. S. (2014). Tracker-level fusion for robust bayesian visual tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, *25*(5), 776–789.

Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2544–2550. https://doi.org/10.1109/CVPR.2010.5539960

Bonin-Font, F., Ortiz, A., & Oliver, G. (2008). Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, *53*(3), 263–296.

Brunetti, A., Buongiorno, D., Trotta, G. F., & Bevilacqua, V. (2018). Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, *300*, 17–33.

Dai, K., Wang, D., Lu, H., Sun, C., & Li, J. (2019). Visual tracking via adaptive spatially-regularized correlation filters. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4670–4679.

Danelljan, M., Bhat, G., Khan, F. S., & Felsberg, M. (2019). Atom: Accurate tracking by overlap maximization. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4660–4669.

Danelljan, M., Bhat, G., Shahbaz Khan, F., & Felsberg, M. (2017). Eco: Efficient convolution operators for tracking. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6638–6646.

Danelljan, M., Gool, L. V., & Timofte, R. (2020). Probabilistic regression for visual tracking. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 7183–7192.

Dunnhofer, M., Martinel, N., & Micheloni, C. (2020). Tracking-by-trackers with a distilled and reinforced model. *Proceedings of the Asian Conference on Computer Vision*.

Dunnhofer, M., Simonato, K., & Micheloni, C. (2022). Combining complementary trackers for enhanced long-term visual object tracking. *Image and Vision Computing*, *122*, 104448.

Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., & Ling, H. (2019). Lasot: A high-quality benchmark for large-scale single object tracking. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5374–5383.

Fu, Z., Liu, Q., Fu, Z., & Wang, Y. (2021). Stmtrack: Template-free visual tracking with space-time memory networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13774–13783.

Haritaoglu, I., Harwood, D., & Davis, L. S. (2000). W/sup 4: Real-time surveillance of people and their activities. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(8), 809–830.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Held, D., Thrun, S., & Savarese, S. (2016). Learning to track at 100 fps with deep regression networks. *European conference on computer vision*, 749–765.

Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2012). Exploiting the circulant structure of tracking-by-detection with kernels. *European conference on computer vision*, 702–715.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Huang, L., Zhao, X., & Huang, K. (2019). Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(5), 1562–1577.

Iqbal, O., Muro, V. I. T., Katoch, S., Spanias, A., & Jayasuriya, S. (2022). Adaptive subsampling for roi-based visual tracking: Algorithms and fpga implementation. *IEEE Access*, *10*, 90507–90522.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Lee, K.-H., & Hwang, J.-N. (2015). On-road pedestrian tracking across multiple driving recorders. *IEEE Transactions on Multimedia*, *17*(9), 1429–1438. https://doi.org/10.1109/TMM.2015.2455418

Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., & Yan, J. (2019). Siamrpn++: Evolution of siamese visual tracking with very deep networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4277–4286. https://doi.org/10.1109/CVPR.2019.00441

Li, B., Yan, J., Wu, W., Zhu, Z., & Hu, X. (2018). High performance visual tracking with siamese region proposal network. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8971–8980. https://doi.org/10.1109/CVPR.2018.00935

Li, Y., & Zhu, J. (2014). A scale adaptive kernel correlation filter tracker with feature integration. *European conference on computer vision*, 254–265.

Lin, L., Fan, H., Xu, Y., & Ling, H. (2021). Swintrack: A simple and strong baseline for transformer tracking. *arXiv preprint arXiv:2112.00995*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *European conference on computer vision*, 740–755.

Lu, X., Ma, C., Ni, B., Yang, X., Reid, I., & Yang, M.-H. (2018). Deep regression tracking with shrinkage loss. *Proceedings of the European conference on computer vision (ECCV)*, 353–369.

Ma, C., Yang, X., Zhang, C., & Yang, M.-H. (2015). Long-term correlation tracking. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5388–5396.

Moorthy, S., & Joo, Y. H. (2021). Multi-expert visual tracking using hierarchical convolutional feature fusion via contextual information. *Information Sciences*, *546*, 996–1013.

Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., & Ghanem, B. (2018). Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. *Proceedings of the European conference on computer vision (ECCV)*, 300–317.

Nam, H., Baek, M., & Han, B. (2016). Modeling and propagating cnns in a tree structure for visual tracking. *arXiv preprint arXiv:1608.07242*.

Nam, H., & Han, B. (2016). Learning multi-domain convolutional neural networks for visual tracking. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4293–4302.

Rais, A. H., & Munir, R. (2021). Vehicle speed estimation using yolo, kalman filter, and frame sampling. *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, 1–6. https://doi.org/10.1109/ICAICTA53211.2021.9640272

Real, E., Shlens, J., Mazzocchi, S., Pan, X., & Vanhoucke, V. (2017). Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5296–5305.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211–252.

Smeulders, A. W., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., & Shah, M. (2013). Visual tracking: An experimental survey. *IEEE transactions on pattern analysis and machine intelligence*, *36*(7), 1442–1468.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Tang, S., Andriluka, M., Andres, B., & Schiele, B. (2017). Multiple people tracking by lifted multicut and person re-identification. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3539–3548.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vojir, T., Matas, J., & Noskova, J. (2016). Online adaptive hidden markov model for multi-tracker fusion. *Computer Vision and Image Understanding*, *153*, 109–119.

Wang, G., Li, N., & Zhang, Y. (2017). Diffusion distributed kalman filter over sensor networks without exchanging raw measurements. *Signal Processing*, *132*, 1–7.

Wang, N., Shi, J., Yeung, D.-Y., & Jia, J. (2015). Understanding and diagnosing visual tracking systems. *Proceedings of the IEEE international conference on computer vision*, 3101–3109.

Wang, N., Zhou, W., Tian, Q., Hong, R., Wang, M., & Li, H. (2018). Multi-cue correlation filters for robust visual tracking. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4844–4853. https://doi.org/10.1109/CVPR.2018.00509

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. *2017 IEEE international conference on image processing (ICIP)*, 3645–3649.

Wu, Y., Lim, J., & Yang, M.-H. (2013). Online object tracking: A benchmark. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2411–2418.

Wu, Y., Lim, J., & Yang, M.-H. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *37*(09), 1834–1848.

Xie, C., Wang, N., Zhou, W., Li, W., & Li, H. (2019). Multi-tracker fusion via adaptive outlier detection. *Multimedia Tools and Applications*, *78*(2), 2227–2250.

Xu, Y., Wang, Z., Li, Z., Yuan, Y., & Yu, G. (2020). Siamfc++: Towards robust and accurate visual tracking with target estimation guidelines. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(07), 12549–12556.

Yan, B., Peng, H., Fu, J., Wang, D., & Lu, H. (2021). Learning spatio-temporal transformer for visual tracking. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10448–10457.

Yonggui, L., & Bugong, X. (2012). Distributed optimal kalman filtering for collaboration estimation in wireless sensor networks. *Proceedings of the 31st Chinese Control Conference*, 6540–6545.

Yoon, J. H., Kim, D. Y., & Yoon, K.-J. (2012). Visual tracking via adaptive tracker selection with multiple features. *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part IV 12*, 28–41.

Zhang, J., Ma, S., & Sclaroff, S. (2014). Meem: Robust tracking via multiple experts using entropy minimization. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, 188–203.

Zhang, W., Song, R., Li, Y., et al. (2020). Online decision based visual tracking via reinforcement learning. *Advances in Neural Information Processing Systems*, *33*, 11778–11788.

Zhang, Y., Wang, T., Liu, K., Zhang, B., & Chen, L. (2021). Recent advances of single-object tracking methods: A brief survey. *Neurocomputing*, *455*, 1–11.

Zhao, Y., & Guo, G. (2017). Distributed tracking control of mobile sensor networks with intermittent communications. *Journal of the Franklin Institute*, *354*(8), 3634–3647.

Zhong, Y., & Liu, Y. (2021). Flexible optimal kalman filtering in wireless sensor networks with intermittent observations. *Journal of the Franklin Institute*, *358*(9), 5073–5088.