UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Master's Degree in ICT Research and Innovation

Image Processing and Computer Vision Program

MASTER THESIS

LONG-TERM TRACKING WITH TARGET RE-IDENTIFICATION

Erik Velasco Salido
Advisor: José María Martínez Sánchez

July 2017

LONG-TERM TRACKING WITH TARGET RE-IDENTIFICATION

Erik Velasco Salido
Advisor: José María Martínez Sánchez

Video Processing and Understanding Lab
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
July 2017

This work has been partially supported by the Spanish Government
TEC2014-53176-R (HAVideo) (2015-2017)

Resumen

El objetivo de este Trabajo Fin de Máster es mejorar el rendimiento del algoritmo de seguimiento de objetos PKLTF (*Point-based* Kanade Lucas Tomasi *colour-Filter*). Para ello, se ha diseñado un algoritmo mejorado en función de las carencias que se han observado en el algoritmo base. Se han propuesto varias mejoras que se han ido implementando sobre el algoritmo base.

Finalmente algunas de ellas se han incorporado al algoritmo propuesto SAPKLTF (*Scale Adaptive Point-based* Kanade Lucas Tomasi *colour-Filter*). Estas mejoras implementadas permiten mejorar el rendimiento frente a los cambios de escala y mantener el rendimiento en tiempo real. Por último, el algoritmo de seguimiento de objetos propuesto se ha evaluado frente a una selección representativa de algoritmos de seguimiento de objetos del Estado del Arte. El nuevo algoritmo de seguimiento de objetos mejora el rendimiento del algoritmo base en la evaluación comparativa, asi como su competitividad frente a los del Estado del Arte.

Palabras clave

Análisis de vídeo, seguimiento de objetos, algoritmos de seguimiento, evaluación de rendimiento, métricas de evaluación, marco de evaluación.

Abstract

The objective of this Master Thesis is to improve the performance of an existing tracker, called PKLTF (Point-based Kanade Lucas Tomasi colour-Filter). A newly improved tracker is designed considering the problems that affect the base tracker. Several improvements are tested, some of which are integrated into the proposed version SAPKLTF (Scale Adaptive Point-based Kanade Lucas Tomasi colour-Filter). These improvements allow to deal with scale changes and maintain the real-time performance. Finally, the proposed tracking algorithm is evaluated against a representative selection of trackers of the state-of-the-art. The new tracker improves the performance of the base tracker in the comparative evaluation, as well as this competitiveness against the ones for the State-of-the-Art.

Keywords

Video analysis, video object tracking, tracking algorithms, performance evaluation, evaluation metrics, evaluation framework.

Acknowledgements

I would like to express my gratitude to my advisor, Chema, for the useful comments, remarks and engagement through the learning process of this Master Thesis. Thanks to Jesús, for making the IPCV possible. Thanks to my friends, VPULab mates, and IPCV classmates, everything is possible on your side. Finally, I would also like to thank my family who helped me to reach the final of this road.

Erik Velasco Salido
June 2017

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Motivation

In Computer Vision one of the most important fields is video object tracking. The application field of video object tracking is very extensive including, among others, video surveillance, augmented reality, medical imaging, traffic control, video editing, etc. Video object tracking is, in general, a time-consuming process due to the huge quantity of information that needs to be taken into account. Development of video tracking algorithms is considered a complex task, with a huge diversity of approaches having been developed in the last years. In this sense, each different scenario has different approaches that have a better performance. Additionally, several difficulties such as occlusions, clutter, illumination changes, and appearance changes, have to be taken into account. All of this makes difficult to obtain a unique algorithm that achieves a perfect performance in all scenarios.

1.2. Objectives

This Master Thesis will study strategies for long-term video tracking based on target update and re-identification in order to improve an existing tracker ,namely, the PKLTF (Point-based Kanade Lucas Tomasi colour-Filter) [1].

The work is divided in four main objectives.

- An evaluation framework will be proposed.
- A selection of video object tracking algorithms, extracted from the state of the art, will be evaluated within the initial mentioned framework.

- Some improvements will be proposed and tested with the aim of improving the initial tracker results, resulting in an improved tracker.
- The improved approach will be integrated in applications in order to demonstrate the operation of the tracker in real situations.

1.3. Document Structure

The structure of the document is as follows:

- Chapter 1: This chapter introduces the work and presents the motivation and the objectives of the Master Thesis.
- Chapter 2: This chapter presents an overview of the literature related to the work presented in this Master Thesis and provides a comparative evaluation of selected trackers in a rigorous evaluation framework.
- Chapter 3: This chapter presents the proposed improvements and the proposed final algorithm.
- Chapter 4: This chapter presents the comparative evaluation results of the proposed tracker.
- Chapter 5: This chapter summarizes the main achievements of the work, discusses the obtained results and provides suggestions for future work..
- References.
- Appendix A: This Appendix contains the comparative evaluation results per sequence of selected trackers.
- Appendix B: This Appendix presents the comparative evaluation results per sequence of the proposed tracker.

Chapter 2

State Of The Art

2.1. Introduction

Object tracking is one of the most important tasks in computer vision, many applications fields such as human-computer interaction, robotics, video-surveillance or augmented reality, among many others, demand reliable and robust target location estimation. Tracking can be defined [2] as "the analysis of video sequences for the purpose of establishing the location of the target over a sequence of frames (time) starting from the bounding box given in the first frame". The performance of tracking algorithms is affected by numerous factors such as illumination changes, fast object motion changes, occlusions, background clutters, etc. These factors make object tracking an open problem when trying to handle with different scenarios using a generic object tracking approach. Therefore, it is important to identify the strengths and weaknesses of tracking algorithms to develop more robust algorithms.

We can divide the tracking approaches into two categories: short-term and long-term tracking. Short-term tracking [3] assumes that the target is visible in the given image and the tracking algorithm estimates the target position in the next image, assuming not disappearance nor complete occlusion of the object. Furthermore, in long-term tracking [4], the target can go away from the field of view and can suffer complete occlusions. Additionally, long-term tracking [4] relates to tracking a target in sequences with a duration larger than 2 minutes, ideally more than 10 minutes.

In the State-of-the-Art, there are many strategies to solve the problem of short-term tracking, each of them focusing on the optimization of the different aspects of the process, such as, for example, speed, precision, and robustness [5]. Nevertheless, none of these methodologies directly addresses post-failure behavior or failure recovery and consequently, can not be used straightforward for long-term tracking.

Short-term trackers without recovery are not proper for long-term tracking problems. Recovery strategies are critical to discover the target after complete occlusion or target disappearance. Short-term trackers, likewise, do not address, generally, the issue of appearance change over the time.

According to [2], a typical short-term visual object tracking system is composed by four modules:

1. Object Initialization.
2. Appearance Modeling.
3. Motion Estimation.
4. Object Localization.

Short-term approaches have, in general, rather poor performances in long-term situations, but are a key element in order to solve the issue of long-term tracking. To solve the problem of long-term tracking, the algorithm must be adapted to appearance changes and occlusions, and should also have the capability to re-identify the target when the object reappears in the scene or occlusion ends.

In the long-term tracking State-of-the-Art, the number of references is very limited in comparison to short-term tracking; this may be because the short-term tracking is still an open problem, and the long-term approaches that have really good performance in terms of accuracy of target tracking are not able to work in real-time.

The long-term situations are more close to real scenarios than short-term, here lies the importance of this kind of approaches: to solve the problem of object tracking in real situations.

2.2. Tracking algorithms

In the State-of-the-Art, there are many algorithms that try to solve the problem of tracking. Most part of these algorithms are focused in short-term tracking. Generally, the most reliable algorithms are those that have a good performance in the most recent challenges: VOT 2015 and 2016[1] [6, 7].

Firstly, we briefly introduce an overview of the basic (classic) tracking algorithms: Lucas-Kanade tracker [8], Particle Filter [9], Kalman Filter [10] and Template Matching [11]. Afterwards, we explain briefly a reduced set of the top performance track-

[1] 5th Visual Object Tracking Challenge (VOT 2017) submission deadline was June 19th 2017. Our tracker satisfies participation requirements and our results were submitted. For more information: http://www.votchallenge.net/vot2017/index.html

ers: Scale Adaptive Mean Shift (ASMS)[12], Edge Box Tracker (EBT) [13], Multi-Domain Convolutional Neural Network Tracker (MDNeT) [14], Continuous Convolution Operator Tracker (C-COT) [15] and Tree-structured Convolutional Neural Network Tracker (TCNN) [16]. Finally, we introduce a selection of long-term tracking algorithms: Tracking-Learning Detection tracking (TLD) [17], Point-based Kanade Lucas Tomasi color-Filter (PKLTF) [1] and Long-Term Featureless Object Tracker (LT-FLO) [18].

2.2.1. Basic trackers

2.2.1.1. Lucas-Kanade tracker

The Lucas-Kanade [8] tracking algorithm is a single-target approach. Firstly, the target **model** is defined as the image inside of the region of interest. After that, the model is searched, performing a **gradient descent**, in the current frame: the parameters of the plane that best aligns the transformed image with the image target are found. This process is performed for each frame until the end of the sequence is reached.

The Lucas-Kanade tracker performs well in situations with small variations, but has problems with occlusions, illumination changes or complex movements.

2.2.1.2. Particle Filter

The Particle Filter [19] approach is used to estimate the state of a system that changes with time. The tracking algorithm can be divided into the following steps: **Create** a target model using the color histogram of the pixels belonging the target region; **Initialize** the samples, in this step the Particle Filter generates a random set of particles over the image, this random set can be created with the previous knowledge or totally random; **Prediction** phase, the state of each sample of the previous frame is slightly modified, e.g., adding noise, to estimate the state of the target in the current frame; **Weighting** phase, each particle in the current frame is weighted according to the similarity of them with the target model; **Sampling** stage, the particles with high weights are replicated, while ones with the low weights are discarded. The steps from prediction to sampling are repeated for each frame of the sequence.

This kind of approach has a good performance in complex scenarios.

2.2.1.3. Kalman filter

The Kalman filter [10] tracker is a single-target tracking algorithm. This algorithm has two phases: **estimation** and **correction**. **Estimation** uses the previous knowledge of the state to estimate the current state. After that, **Correction** uses the present measurement, such as target location, to correct the state.

The Kalman filter tracker works well when the target is shifted with constant velocity or constant acceleration.

2.2.1.4. Template Matching

The Template Matching [11] tracking algorithm is a single-target approach. The main idea behind this approach is to model the target as the sub-image inside of the region of interest, that will be used as a template. This template is searched in each frame by comparing pixel intensities. The algorithm is divided into the following steps: **Selection** of the target; **Computation** of the convolution or a sum-comparing metric, as sum of squared differences (SSD), or cross-correlation, over the next frame; **Estimation** of the target position that corresponds to the maximum value of the convolution, this maximum corresponds to the center of the estimated target and the estimated size of the target is the model size; **Repetition** of the loop to estimate the target in the next frame.

This tracking algorithm is simple and has a good performance in low complexity scenarios.

2.2.2. Short-term trackers

2.2.2.1. Scale Adaptive Mean Shift

ASMS tracker [12] is based on Mean Shift [20] introducing a modification to deal with the issue of scale adaptation with an innovative method to estimate the scale. This method also introduces changes in the way to improve the robustness in the scale estimation with background clutter. For this purpose, a forward-backward checking and a weighted color histogram are used in the algorithm. This tracker is able to work in real-time with a relatively good performance in terms of accuracy.

2.2.2.2. Edge Box Tracker

EBT [13] uses object proposals, based on edges in the detection process, to track the target. With this approach based on contours, the computational cost of finding the target is reduced, allowing to perform the search in the whole image but also

enables to focus on the best candidates to test and upgrade the model. The best candidates are employed to split into positive and negative samples in order to improve the tracking performance. This tracker uses the Structural Support Vector Machines classifier [21] that permits online updating. To describe the target, it employs 16-bin RGB intensity histograms from a 5-levels spatial pyramid. This approach is faster than CNN approximations, but still remains slow to work in real time.

2.2.2.3. Multi-Domain Convolutional Neural Network Tracker

MDNeT [14] uses a CNN to represent the target object. This CNN is pre-trained with 80 sequences, and their corresponding ground-truths annotations, to generate a general model that can deal with new different sequences. The network is divided in two kind of layers: shared layers and domain specific layers. In the shared layers, a generic representation is obtained after training. The domains layers are associated with the individual tracking sequences and every domain has a different branch for binary classification. To track the target, the sample with the highest score among the candidates obtained around the previous target is searched. This tracker has a good performance in accuracy and robustness, but it is very expensive in time.

2.2.2.4. Continuous Convolution Operator Tracker

C-COT [15] is a single-target tracker based on discriminative correlation filters and introduces a novel approach for training continuous convolution filters. This tracker learns a discriminative continuous convolution operator as its tracking model. The novel learning technique proposed by this work enables an efficient fusion of multi-resolution feature maps; in the case of this tracker, it is possible to use features of pre-trained Convolutional Neural Networks. The tracker performs very well in accuracy as well as robustness, but is far away from working in real-time.

2.2.2.5. Tree-structured Convolutional Neural Network Tracker

TCNN[16] is a single-target tracker based on Convolutional Neural Networks (CNNs) and has two main steps: state estimation and model update. This tracker uses CNNs with multiple target appearance models in a tree structure to maintain the model consistency and handle with appearance changes effectively. The workflow of this tracker is the following: for each new frame, candidate samples are taken around the target estimated in the previous frame, and the likelihood of each sample is determined by the weighted average of the scores from manifold CNNs; for each CNN, the weight is calculated according to the reliability of the path along which the tree

structure has been updated; the maximum likelihood candidate sample is selected as the new target state in the present frame; finally, a new CNN is generated from the previous ones after a fixed number of frames and it is assigned the greater weight in the estimation of the state. As in the case of C-COT tracker, the performance is good in terms of accuracy and robustness, but is heavy in computation time, which limits its practical applicability.

2.2.3. Long-term trackers

2.2.3.1. Tracking-Learning Detection

TLD [17] is a single-object long-term tracker. This tracker is a combination of tracking and detection with online model learning. Under the assumption that the target is visible and the motion is limited along consecutive frames, the target position is estimated. The detector searches exhaustively in every frame the target model. With the combination of the tracking location and the detector, the samples, positive and negative, are generated to learn the appearance of the model in order to avoid false detections. The idea in the learning process is to be able to identify the errors and correct them. This tracker has a good performance when it has time to learn a correct model in the first frames. In terms of computational cost, it is relatively moderate, but without getting real-time performance.

2.2.3.2. Point-based Kanade Lucas Tomasi color-Filter

PKLTF tracker [1] is a single-target tracker focused on long-term situations, that is able to recover the target after a loss. This tracker is robust to occlusions and appearance changes. The tracking process is divided into two main steps. The first phase consists of using the Kanade Lucas Tomasi approach (KLT) to search the object features, while the second one is the application of the Mean Shift gradient descent approach to estimate the target position. The object model employed is based on the combination of RGB and luminance gradient in the form of a histogram. It achieves good results in long-term situations, with the advantage that this tracker is capable of working in real-time.

2.2.3.3. Long-Term Featureless Object

LT-FLO [18] is a long-term tracking algorithm. This tracker uses edge points and if the stability of the gradient in these points is not consistent between frames, the tracker detects the object disappearance. The candidates are all correspondences between the local maxima of the gradient magnitude and the tangent lines to the

edges. After that, the candidates are used to compute a similarity transformation with RANSAC, frame by frame. If the confidence of the estimation is low, the tracker resets the position estimated to the previously known position. Furthermore, the state is learned if the confidence is high for future rectifications. The performance of this tracker in short-term situations [6, 7] is not remarkable but has a decent throughput in long-term scenarios. In terms of time-consuming, it is far away from real-time.

2.3. Datasets

This section introduces the most important datasets in the State-of-the-Art in short-term and long-term tracking. These datasets have been used in the most important object tracking challenges.

The Visual Object Tracking (VOT) [3, 6, 7, 22] is the reference challenge in short-term tracking; this competition started in the year 2013 and is celebrated each year with a huge number of tracking algorithms submitted. Further important datasets for short-term tracking scenarios are the Object Tracking Benchmark (OTB) [23] and the NUS People and Rigid Objects (NUS-PRO) [24].

In long-term tracking, the Long-Term Detection and Tracking (LTDT) challenge [4] celebrated in 2014 has a public dataset. The last dataset available, is also a long-term dataset, the Teacher Tracking dataset (TTds) [1].

2.3.1. Visual Object Tracking 2016 dataset

The VOT 2016 [6] dataset has 60 sequences that are the same as the ones of the VOT 2015 [6] dataset; each sequence is per-frame annotated and these annotations have six different visual attributes: camera motion, size change, illumination change, occlusion, motion change or unassigned if the frame can not be included in any of the previous categories. The ground-truth bounding boxes were generated with an automatic tool that uses a segmentation mask. This bounding box contains the maximum number of foreground pixels, with the lower number of background pixels.

2.3.2. Object Tracking Benchmark dataset

The OTB [23] dataset is another relevant dataset in the State-of-the-Art of short-term tracking; this dataset has 100 sequences divided into two sub-sets of 50 sequences, most of them are included in the VOT 2016 [7] dataset. In this dataset, the annotation classifies the frames with 9 different attributes, and has ground-truth bounding boxes available for all sequences.

2.3.3. NUS People and Rigid Objects dataset

The NUS-PRO[24] dataset is composed of 365 sequences taken from YouTube. These videos are classified into five different classes: face, pedestrian, sportsman, rigid object, and long sequences. The duration of the sequences is between 6 seconds to 2 minutes and 48 seconds. This dataset has ground-truth bounding boxes available for all sequences, and also has annotated 12 categories including shadow change, flash, rotation, shape deformation, scale change, dim light, clutter background, fast background change, partial occlusion, full occlusion, similar objects and camera shake.

2.3.4. Long-Term Detection and Tracking dataset

The LTDT [4] dataset has 6 sequences with a duration between 1 minute and 10 minutes. All the sequences have ground-truth bounding boxes; in this case, if the object is occluded more than 50% or the object disappears from the field-of-view bounding box are not annotated.

2.3.5. Teacher Tracking dataset

The TTds [1] is a dataset with 12 sequences, 2 of them have a duration of 30 minutes and the other 10 are shorter in duration, between 10 seconds and 1.10 minutes. These shorter sequences are focused on more challenging situations. This dataset doid not have ground-truth, therefore, it was generated in this work

2.4. Evaluation framework and metrics

2.4.1. Tracking-st

In order to evaluate the performance of the trackers we use the evaluation framework, tracking-st [25] , developed in the VPU-Lab. In this framework, we can load multiple datasets and trackers at the same time, and automatically compute different metrics to analyze the performance of the trackers.

The tracking-st framework [25] provides an easy and automatic evaluation of single-object video trackers. This framework allows a simple testing of trackers in batch or parallel mode, and also gives support for GPU-enabled trackers. Another important feature is the standardization of the integration of new trackers. Lastly, the framework is capable of automatically downloading the dataset to perform the evaluation.

2.4.2. Metrics

This subsection describes a selection of visual single-object tracking metrics. The ground-truth annotation is defined as Λ^G and the estimated target annotation as Λ^T .

2.4.2.1. Spatial overlap

The spatial overlap measures the percentage of overlap between the ground-truth bounding box and the bounding box estimated by the tracker. The spacial overlap $\Phi(\Lambda^G, \Lambda^T)$ is defined [26, 27] with the following equation:

$$\Phi(\Lambda^G, \Lambda^T) = \frac{R_t^G \cap R_t^T}{R_t^G \cup R_t^T} = \frac{TP}{TP + FP + FN} \tag{2.1}$$

where R_t^G, R_t^T denotes, respectively, the region of the object at time t for the ground-truth and the estimated target; TP are the true positive pixels; FP are the false positive pixels; and FN are the false negative pixels.

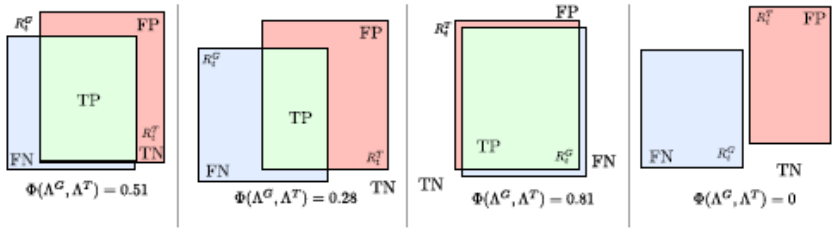


Figure 2.1: Different spatial overlap situations [27]

2.4.2.2. Center error

The center error (CE) [27] is defined as the difference in number of pixels between the estimated center position of the bounding box predicted by the tracker and the center of the ground-truth bounding box. The center error $\triangle(\Lambda^G, \Lambda^T)$ follows the equation 2.2

$$\triangle(\Lambda^G, \Lambda^T) = \{\delta_t\}_{t=1}^N, \ \delta_t = ||x_t^G - x_t^T||. \tag{2.2}$$

where x_t^G, x_t^T are, respectively, the center position at the time t for the ground-truth and the predicted target; and N is the duration of the sequence.

2.4.2.3. Sequence Frame Detection Accuracy

The sequence frame detection accuracy (SFDA) [26] is a metric that contains information regarding the spatial overlap, missed detections, false positives and the number of detected targets. The SFDA provides a ratio of the spatial intersection and union between two object locations. The followings equations define the SFDA metric:

$$SFDA = \frac{\sum_{t=1}^{t=N} FDA(t)}{\sum_{t=1}^{t=N} \exists(N_t^G \, OR \, N_t^T)} \tag{2.3}$$

$$FDA(t) = \frac{overlap\_ratio}{\frac{N_t^G + N_t^T}{2}} \tag{2.4}$$

$$overlap\_ratio = \sum_{i=1}^{N_t^{mapped}} \frac{|R_t^{G(i)} \cap R_t^{T(i)}|}{|R_t^{G(i)} \cup R_t^{T(i)}|} \tag{2.5}$$

where $R_t^{G(i)}$, $R_t^{T(i)}$ denote, respectively, the *i-th* object in the frame t, for the ground-truth and the detected object; N_t^{mapped} is the number of matched pairs of ground-truth annotation and estimated target location in frame t; N_t^G, N_t^T are the number of the ground-truth and predicted targets in frame t; and N is the number of frames in the sequence.

2.4.2.4. Average Tracking Accuracy

The average tracking accuracy (ATA) [26] measures the accuracy of the tracking, penalizing fragmentation in both the spatial and the temporal domains. ATA is defined by the following equations:

$$ATA = \frac{STDA}{\frac{N^G + N^T}{2}} \tag{2.6}$$

$$STDA = \sum_{i=1}^{N^{mapped}} \frac{\sum_{t=1}^{N} \frac{|R_t^{G(i)} \cap R_t^{T(i)}|}{|R_t^{G(i)} \cup R_t^{T(i)}|}}{M_{(R^{G(i)} \cup R^{T(i)} \neq 0)}} \tag{2.7}$$

where $R_t^{G(i)}$, $R_t^{T(i)}$ denote, respectively, the *i-th* object in the frame t, for the ground-truth and the detected object; N^{mapped} is the number of matched pairs of ground-truth annotation and estimated target location; N^G, N^T are the number of the ground-truth and predicted targets; and N is the number of frames in the sequence.

2.4.2.5. Track Completeness

The track completeness (TC) [26] measures the ratio between the time where the overlap between the estimated and the ground-truth target is bigger than a threshold, and the ground-truth duration. The following equation defines TC:

$$TC = \frac{\sum_{t=1}^{N_T^t} O(R_t^G, R_t^T)}{N_G} \tag{2.8}$$

where $O(R_t^G, R_t^T)$ is a binary value that is 0 when the overlap is less than a threshold, and 1 otherwise; and N_T^t, N_G are, respectively, the track duration for the estimated object and the ground-truth.

2.4.2.6. Combined Tracking Performance Score

The combined tracking performance score (CoTPS) [26, 27] combines the information of tracking accuracy and tracking failure in a single score. CoTPS is defined by the following equation:

$$CoTPS = 1 - \Phi - (1 - \lambda_0)\lambda_0 \tag{2.9}$$

where $\Phi$is the average overlap; and λ_0 is the percentage of frames where the overlap is 0.

2.5. Selected trackers, datasets and metrics: a comparative evaluation of State-of-the-Art

In this section, we justify the selection of the trackers, dataset and metrics that are used in our work and present a comparative evaluation of the selected trackers in the datasets.

2.5.1. Trackers

The trackers selected for the comparative performance evaluation are the followings:

- The C-COT [15] tracker is selected because it was the top performance tracker in the VOT2016 challenge [7].
- The MDNET [14] tracker is selected because was the top performance tracker in VOT2015 challenge [6].

- The ASMS [12] tracker is used in our work because it has a relative good performance and works in real-time.
- The TLD [17] tracker is selected because it is a long-term tracker with a good performance in both scenarios, short-term and long-term.
- The LT-FLO [18] is included in the work to have another long-term tracking comparative reference, despite its discreet performance.
- The PKLTF [1] has a good performance in long-term scenarios and it works close to real-time, for these reasons it is included in the comparative.

The trackers not included are: the TCNN [16] tracker and the EBT [13] tracker, because the performance of these is worst than the other short-term trackers included in the comparative.

2.5.2. Datasets

The comparative evaluation are performed with this selection of datsets:

- The VOT 2016 [7] dataset is selected because it is the reference in short-term tracking evaluation.
- The LTDT 2014 [4] dataset is included because it is the only one available in long-term tracking with ground-truth.
- The TTds [1] dataset is selected because it is the dataset with longer sequences.

The OTB [23] dataset is excluded because a huge part of the sequences are included in the VOT 2016 dataset, and the NUS-PRO [24] dataset is also excluded because many of the situations included in this dataset are similar to the ones in VOT 2016.

2.5.3. Metrics

In the case of single-object tracking most of the metrics described before have a big correlation between them, as the works [26, 27] demonstrated. The metrics selected to perform the comparative evaluation are:

- The Spatial Overlap Φ, because it is less correlated with the Center Error (CE) than CoTPS as shown in Figure 2.2.
- The Center Error is the least correlated with the other metrics.

- The processing time per frame is measured in the comparative evaluation, because real-time working is a key factor in this work.

The SFDA, ATA, TC and CoTPS are highly correlated with Φ(see Table 2.1), therefore we choose only this metric among all of them.

It must be notified, that depending on the final application the selected metrics may be more or less relevant. For example, if real-time is key, processing time is more relevant, whilst for wide-range tracking CE is more important.

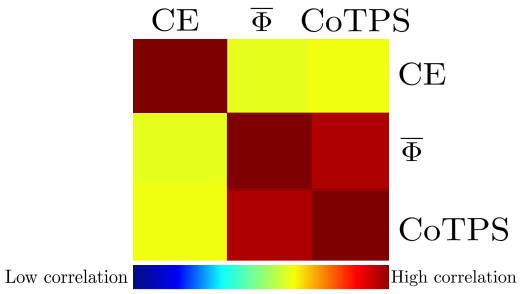


Figure 2.2: Correlation between CE, Φ and CoTPS [27]

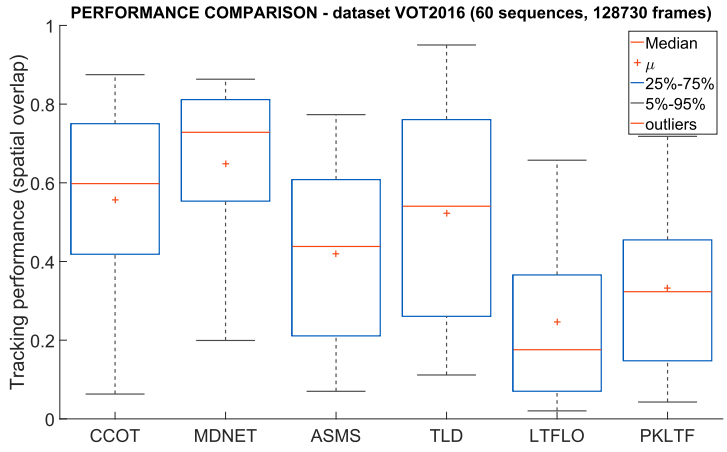
Table 2.1: Correlation between the metrics [26]

	SFDA	ATA	Φ	TC	CoTPSi
SFDA	1.00	0.99	0.96	0.89	0.88
ATA	0.99	1.00	0.96	0.89	0.88
Φ	0.96	0.96	1.00	0.89	0.92
TC	0.89	0.89	0.89	1.00	0.78
CoTPSi	0.88	0.88	0.92	0.78	1.00

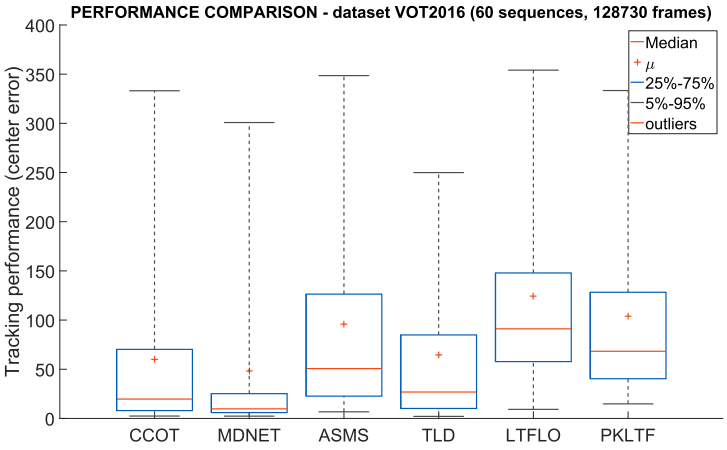
2.5.4. Comparative evaluation results

2.5.4.1. VOT 2016

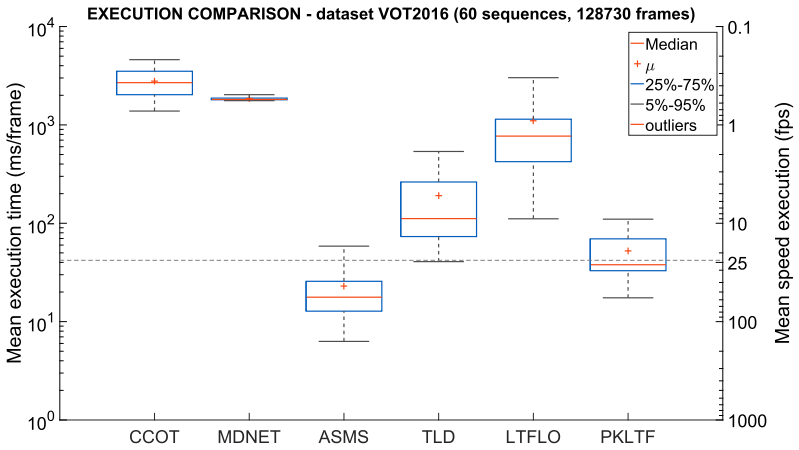
In the VOT 2016 [7] dataset, the top performance tracker is the MDNET [14], this tracker outperforms the others in both metrics (see Figures 2.3 a) and b)). In terms of execution time (see Figure 2.3 c)), only two trackers, ASMS [12] and PKLTF [1], can perform in real-time, and the remaining trackers are far away from working in real-time, with speeds close to 1 frame per second (fps). Between the two trackers that can work in real-time, the ASMS [12] outperforms PKLTF [1]; this performance gap is bigger in the Spatial Overlap than in Center Error, where the difference is small. That is because the PKLTF [1] tracker does not perform a scale adaptation. See Appendix A.1 for the detailed results.



a)



b)



c)

Figure 2.3: VOT 2016 Performance Comparison. a) Spatial Overlap; b) Center Error; c) Execution Time

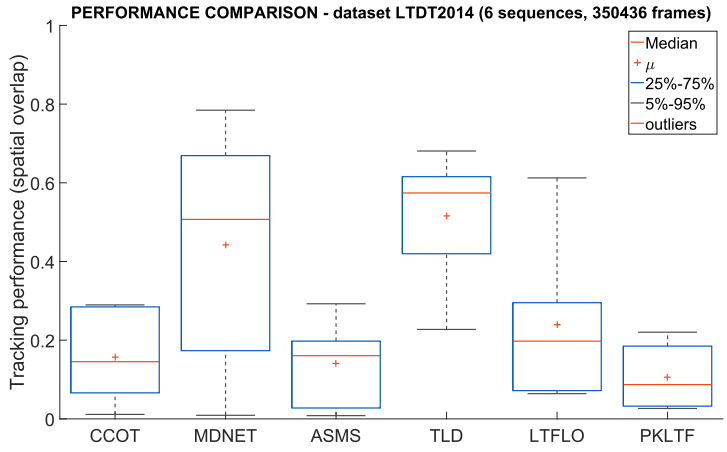
2.5.4.2. LTDT 2014

In the LTDT 2014 [4], the TLD [17] tracker has the best performance, the throughput difference between the TLD and the others is bigger in the Center Error (see Figure 2.4 b)), than in Spatial Overlap (Figure 2.4 b)). The fall of performance of the C-COT [15] tracker is remarkable, the bigger between the short-term trackers. In the execution time comparison (see Figure 2.4 c)), as in the VOT 2016 [7] dataset only two trackers, ASMS [12] and PKLTF [1], are capable to work in real-time. The TLD tracker is close to real-time performance, and the other three are far away from it. Appendix A.2 shows the detailed results.

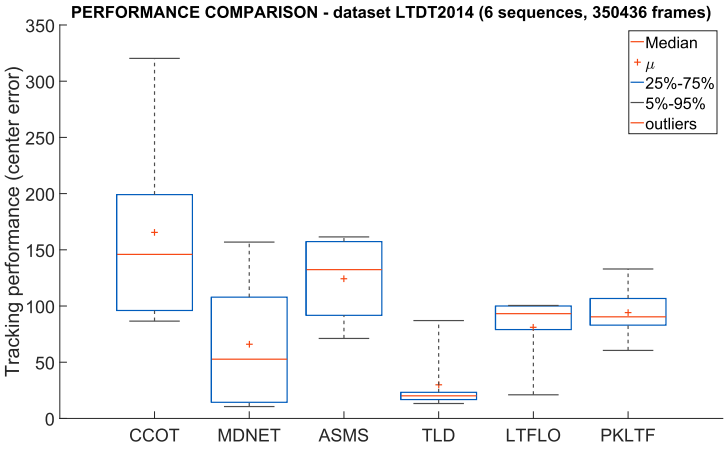
2.5.4.3. TTds

In the TTds [1] dataset, all trackers results are close, except LT-FLO [18] that has the worst performance. In the Spatial Overlap and Center Error, the PKLTF [1], ASMS [12] and C-COT [15] are the best (see Figure 2.5 a) and b)). Moreover, PKLTF and ASMS trackers work in real time, it should be noted that ASMS performs close to 60 fps (see Figure 2.5 c)). Like in the previous datasets, the others trackers are far away from real-time. Detailed results can be found in Appendix A.3.

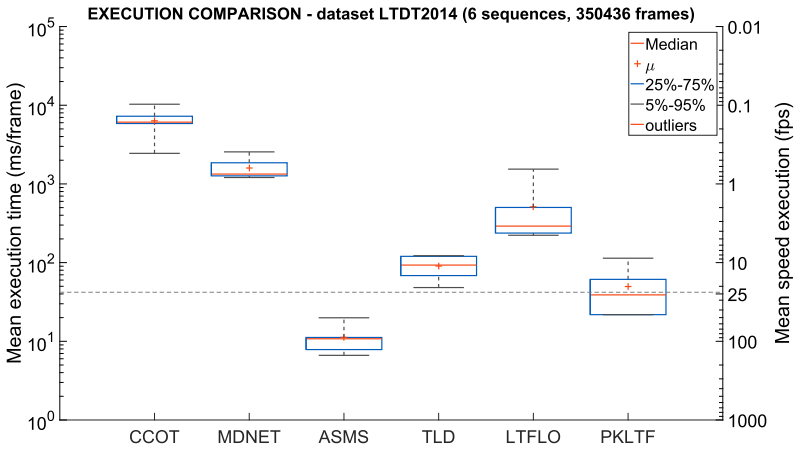
The comparative evaluation shows that not all the trackers are capable to work with the same performance in short-term and long-term scenarios. For example, C-COT [15] fails in long-term scenarios. Also these results show that the real-time execution is only achieved by ASMS [12] and PKLTF [1] trackers.



a)

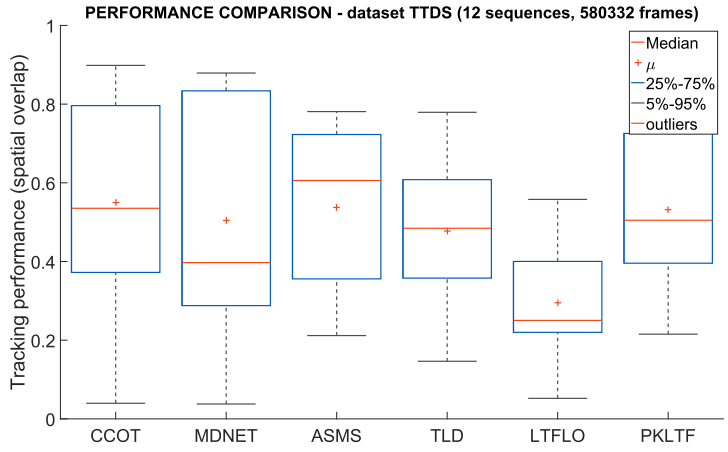


b)

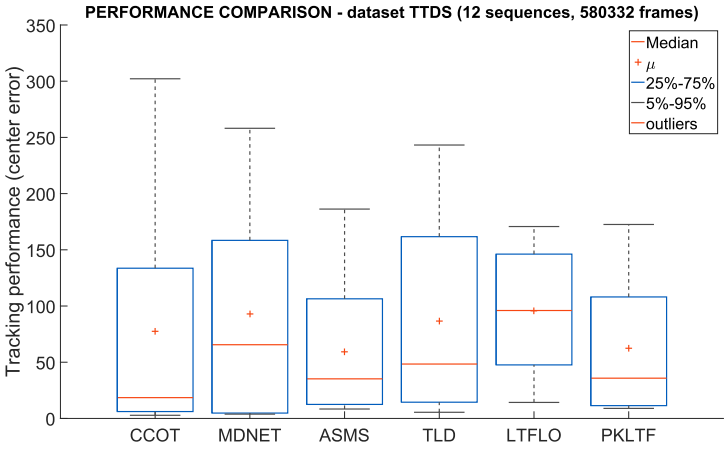


c)

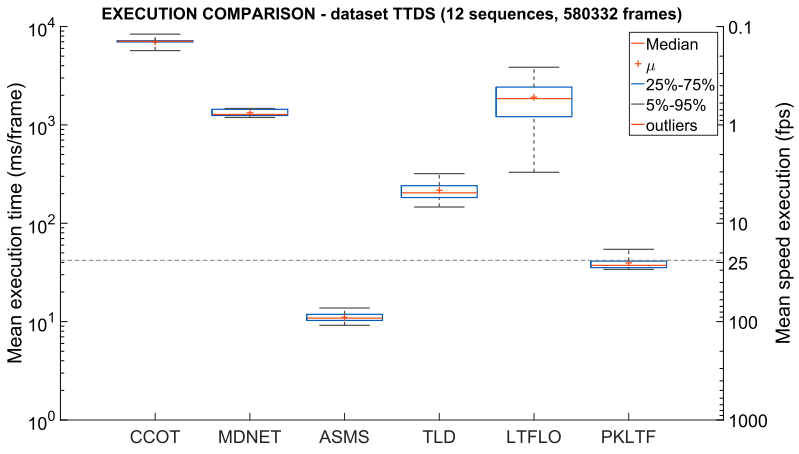
Figure 2.4: LTDT 2014 Performance Comparison. a) Spatial Overlap; b) Center Error; c) Execution Time



a)



b)



c)

Figure 2.5: TTds Performance Comparison. a) Spatial Overlap; b) Center Error; c) Execution Time

2.6. Conclusions

This chapter has presented different tracking approaches, for short-term and long-term situations. Also, it has shown the datasets for both scenarios and the evaluation metrics. Lastly, a performance comparison over the selected datasets with the chosen algorithms has been presented. In this analysis, it is shown that each tracking algorithm has its advantages and disadvantages, depending on the environment in which it runs and its capacities of adaptation to different scenarios.

Chapter 3

Design and development of the SAPKLF tracker

3.1. Introduction

This chapter details the design and development of the Scale Adaptive Point-based Kanade Lucas Tomasi color-Filter (SAPKLTF) tracker, based on the PKLTF one [1]. After a detailed description of PKLTF, the proposed enhancements are justified and described. Finally, the SAPKLTF is described in detail with the changes that we have introduced with respect to the PKLTF tracker.

3.2. Base algorithm

PKLTF [1] is a tracker that supports high appearance changes in the target, occlusions, and is also capable of recovering a target lost during the tracking process. This tracker has an image stabilization module that extracts the feature points between consecutive frames and estimates a homography between them to compensate possible small camera motion. After that, a two stages algorithm has been designed for this single-target object tracker. The first phase is based on the Kanade Lukas Tomasi approach (KLT) [28] to estimate the target position, allowing tracking relatively big displacements. The second step performs a mean shift gradient descent [20] in order to refine the estimation previously done by the KLT and places the target in the correct location. The target model consists of a histogram including the values of the color components RGB and an edge binary flag. Besides, the color model is updated adding weight to the pixels present in the original histogram. Figure 3.1 shows the block diagram of the algorithm.

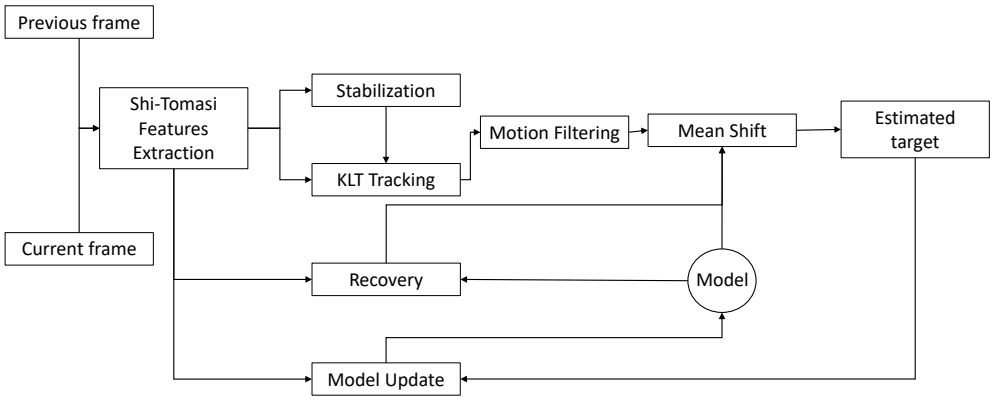


Figure 3.1: PKLTF Architecture

3.2.1. Initialization

The tracking algorithm is initialized only with the bounding box that defines the target in the first frame. The target model is based on the RGB color and on the edge information (RGBE). The model is a one-dimensional histogram that includes the quantized values of the color components (16 bins per color) and the edge binary flag. This histogram is composed by the information of all pixels that are contained inside the bounding box that defines the target in the first frame. The contribution of these pixels are equal for all of them, independently of their position in the bounding box.

3.2.2. Video stabilization

In order to reduce the effect of the possible camera movements, this tracker has a stabilization module. To perform the stabilization, for each input frame the homography between two consecutive frames is estimated, using the Shi-Tomasi [28] features also used for the target tracking in the KLT tracking step. The homography is computed with the RANSAC method. After that, the current frame and the target position are corrected using the homography.

3.2.3. KLT tracking

The first stage of the tracking is performed using a KLT [28] feature tracking. This technique is based on characteristic points tracking, using the optical flow equations developed by Lucas-Kanade, and the iterative Newton-Raphson method for searching the object position. In this case, the tracker uses the Shi-Tomasi corner detector. These features present high repeatability and low computational cost, that enable a quick tracking with a huge number of features. The Shi-Tomasi features avoid the spurious corner points on smooth curves and are invariant to typical image transformations. These features are also used in the stabilization module.

The work flow in this stage is the following: Firstly, the extraction of the Shi-Tomasi features are performed for each frame; Secondly, the movement is estimated in the next frame computing the minimization error process with the features and its displacements; After that, the features are classified in foreground features or background features depending on the motion map computed between the current stabilized frame and the previous frame; Finally, the total displacement of the target is computed with the weighted displacements obtained for each foreground point feature, these weights are dependent on the distance of each point with respect to the center.

In this approach, the KLT tracking method is applied in a pyramidal form; this approach permits to deal with large displacements of the target.

3.2.4. Mean Shift

After the KLT tracking step, in order to refine the estimation of the target position, a Mean Shift [20] is computed. In the case of PKLTF, the Mean Shift is performed using the target model that is built in the initialization step. The Mean Shift technique finds the maximum in the confidence map resulting from comparing the target model to a searching area around the location estimated with the KLT. To compute the confidence, the Bhattacharya distance between histograms is used.

3.2.5. Model update

This tracker updates the target model when the target is estimated, using the location of the Shi-Tomasi features for this update. The histogram is updated increasing in 1 the value of the bin (each bin is defined by the following four values: the values of RGB components and the edge binary flag) associated to each Shi-Tomasi point.

3.2.6. Recovery

The PKLTF tracker is able to recover the target after a loss. The recovery process is activated after a fixed number of consecutive frames (30 frames in this implementation) where the algorithm does not match any characteristic point of the target and the similarity between the histograms is below the defined threshold (0.6 in this implementation). When the recovery process is activated, the Shi-Tomasi features of the current frame are extracted, and over each feature point the similarity between the target model and the histogram of this region is computed. After that, the object is considered as recovered if the similarity between the candidate histogram and the initial target model reach the determinate threshold (0.1 in this implementation), and the tracking process continues with the recovered target. Conversely, the recovery process continues in the following frame if the similarity is below the threshold for all the candidates. This step has a heavy computational cost in comparison with the tracking when the target is not considered lost.

3.3. Improvement proposals

3.3.1. Corrected Background Weighted Histogram

In the PKLTF tracker, the initialization of the object is done with a rectangular bounding box. The problem with this kind of initialization is that it include information of the scene background in the target model, and this increases the probability of drifting during the tracking process. The Corrected Background Weighted Histogram [29] (CBWH) method tries to minimize the impact of the background in the target model.

The CBWH technique consists of: Firstly, the histogram of the bounding box containing the target is calculated, that it is called foreground histogram; After that, a second bounding box centered at the same point as the previous one is generated, but with twice the size of the original; Finally, and in the same way, its histogram is computed, which is denominated background histogram. Once the histograms are obtained, the histogram of the foreground is corrected. To do this, the normalized background histogram is inverted and multiplied by the normalized foreground histogram, so that the background components are reduced in the foreground histogram, maintaining the information that represents the target. This is because the background weight in the background histogram is much larger than that of the target. Therefore, by multiplying by the inverse of the background histogram, the background zone in the final foreground histogram is heavily reduced.

3.3.2. Scale Adaptive

The PKLTF tracker is not able to deal with scale changes and after analyzing its performance in the proposed evaluation framework the need of scale adaptation was concluded. The approach adopted to deal with scale changes is the technique used by the ASMS [12] tracker. In order to implement this technique in the proposed tracker, the CBWH described in the previous section was replaced with the approximation, share the same approach, that introduces the ASMS tracker, called Background Ratio Weighting (BRW).

The BRW method is based on a ratio maximization. The background histogram is calculated over the neighborhood of the target in the first frame as in the CBWH method and the ratio (R) is defined as the Bhattacharya coefficient of the candidate target histogram and the model histogram, divided by the Bhattacharya coefficient of the candidate target histogram and the background histogram. With this ratio, the weights of the background are computed using a gradient ascent method for a maximization of log(R). Once the weights of the background are computed, the process is the same as in the CBWH but using these weights to weight the background histogram.

The scale factor is estimated during the Mean Shift process. The equations that define the scale factor process estimation are the following:

$$h_1 = \left(1 - \frac{w_k}{M_0}\right) h_0 + \left(\frac{w_g}{h_o \cdot M_0}\right) + rs + rb \tag{3.1}$$

$$h_0 = 0.7h_0 + 0.3h_1 \tag{3.2}$$

where w_k is the sum of all weights of the candidate region multiplied by the Epanechnikov Kernel; w_g is the sum of all weights of the candidate region multiplied by the first derivative of Epanechnikov Kernel and the distance respect to the center; M_0 is the sum of all weights of the candidate region by the first derivative of Epanechnikov Kernel; and rs, rb are two regularization terms, the first is related to not drastically change scale and the second one forces to include background pixels in the search window. For the detailed formulation we refer the interested reader to [12].

In addition, the ASMS approach has a process to check the scale estimation, that is called Backward Scale Consistency Check. This consists on backward tracking to validate the estimated scale; in case of scale inconsistency the object size is estimated with a combination of the size in the previous frame, the estimated size and the size in the first frame. In our implementation the parameters that weighting these sizes

a) b)

Figure 3.2: Points features: a) without spatial filtering and b) with spatial filtering.

are the originals proposed for the ASMS tracker.

3.3.3. Spatial Filtering

Another problem detected in the base algorithm is related to the features used to compute the KLT tracking. This features, as has been explained in the previous section, are divided into foreground and background features depending on the motion map (see Figure 3.2 a)). The problem arises when the camera motion can not be stabilized correctly: in this case, many features from the background are classified as foreground features. Another problematic situation is when the target is static and the features in the target region are classified as background. In order to solve this problem, the filtering of the features depending on their distance to the center of the target has been introduced, classifying only as foreground features those contained in the circumference whose diameter is equal to the diagonal of the bounding box target (see Figure 3.2 b)).

3.3.4. Constrained displacement

The constrained displacement is introduced in order to reduce large variations in location estimation. The position variation can be assumed small between consecutive frames. The displacement information in the previous frames can then be used to limit the maximum displacement that can be produced in the estimation of the target position in the current frame. This limitation is defined by the previous displacement plus a margin of tolerance, this allows to handle speed and acceleration changes.

3.3.5. Adaptation to Orientation

The idea used to adapt to the orientation changes is the same that in the Continuously Adaptive Mean Shift (CAMSHIFT) algorithm [30]. This consists in the computation of the second order moments of the mass center and use them to es-

timate the orientation and the size of the target. The Mean Shift algorithm only computes the zeroth moment, M_{00}, and the first moment for x, M_{10} , and y, M_{01}; these moments are defined by the following equations:

$$M_{00} = \sum_x \sum_y I(x, y) \tag{3.3}$$

$$M_{10} = \sum_x \sum_y xI(x, y) \tag{3.4}$$

$$M_{01} = \sum_x \sum_y yI(x, y) \tag{3.5}$$

where $I(x, y)$ is the probability value at position (x, y) in the image. With the zeroth and the first order moments, the centroid can be computed as follow:

$$x_c = \frac{M_{10}}{M_{00}};\ y_c = \frac{M_{01}}{M_{00}} \tag{3.6}$$

The second order moments follow these equations:

$$M_{20} = \sum_x \sum_y x^2 I(x, y) \tag{3.7}$$

$$M_{02} = \sum_x \sum_y y^2 I(x, y) \tag{3.8}$$

$$M_{11} = \sum_x \sum_y xyI(x, y) \tag{3.9}$$

where $I(x, y)$ is the probability value at position (x, y) in the image. With the zeroth and the second order moments, the orientation of the major axis is computed as follows:

$$\theta = \frac{arctan\left(\frac{2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2\right) - \left(\frac{M_{02}}{M_{00}} - y_c^2\right)}\right)}{2} \tag{3.10}$$

Furthermore, the size of the target can be estimated with an an-isotropic scale variation. In this case, the length and the width are computed as follows:

$$l = \sqrt{\frac{(a + c) + \sqrt{b^2 + (a - c)^2}}{2}} \tag{3.11}$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}} \tag{3.12}$$

with a, b, and c defined as:

$$a = \frac{M_{20}}{M_{00}} - x_c;\ b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right);\ c = \frac{M_{02}}{M_{00}} - y_c \tag{3.13}$$

With equations 3.6, 3.10, 3.11 and 3.12 the size and the position of the target bounding box is estimated .

3.4. Improved algorithm

The SAPKLTF tracker is the improved version of PKLTF. The main changes included in the final algorithm are: removing the video stabilization and motion filtering; inclusion of spatial filtering, the constrained displacement, the scale adaptive Mean Shift and the BRW, and the modification of the model re-initialization (update). Figure 3.3 shows the block diagram of the algorithm.

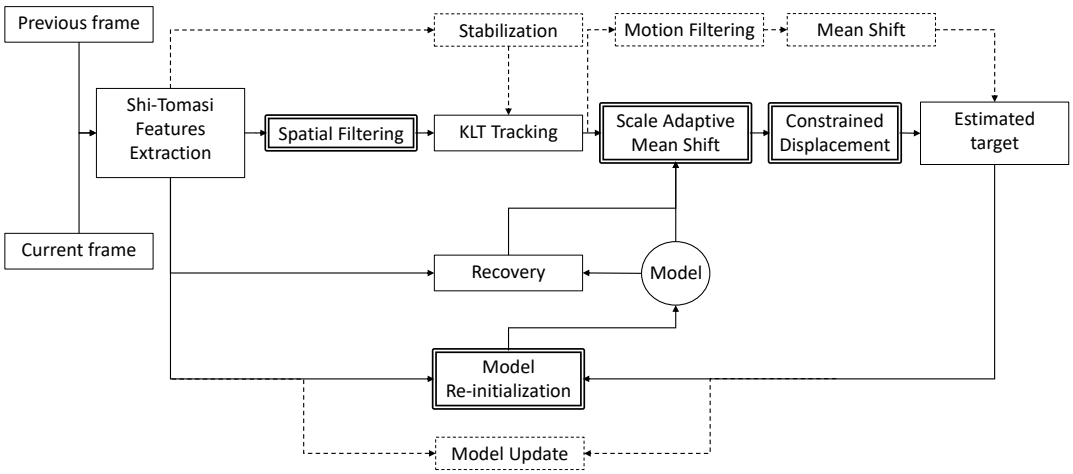


Figure 3.3: SAPKLTF architecture

The video stabilization is removed because is computationally expensive and the problems that generate this module in the following phases when the stabilization is not good.

The motion filtering is also removed because after the elimination of the video stabilization this filtering is noisy, producing incorrect classification between foreground and background features.

The spatial filtering, that in the previous steps was designed to remove outlier features, after removing the motion filtering takes a capital importance because this step realizes now the filtering of features. Also, the constrained displacement plays a relevant role, because it takes advantage of the motion knowledge and limits the possible drifting situations in the Mean Shift refinement step.

The most important change is the scale adaptation and the BRW; these improvements make the estimation of the target more accurate, that also implies better performance in robustness because in the updating step the target is better defined.

The model re-initialization (update) is performed in a different way, but the conditions to update the model are the same than in the base algorithm. The new updating process consists in re-initializing the model with the target of the current frame.

Finally, the adaptive orientation is not included in the improved version because this (expected) improvement performed quite differently in different cases. Additionally, this can not be evaluated with all the selected datasets (only the VOT 2016 dataset has rotated bounding boxes). For the VOT 2016 sequences, the proposed improvement worked correctly for rigid objects, whilst for non-rigid object its performance was worst and quite dependent on the ground-truth annotations.

3.5. SAPKLTF Applications

3.5.1. Demonstrator

The demonstrator application is developed in order to show the operation of the tracker in real situations, to facilitate the understanding of the algorithm and the influence of the parameters on the algorithm performance.

In this case, the PKLTF demonstrator developed in a previous work [31] is used as starting point. This application has been updated with the SAPKLTF tracking algorithm.

The application gives a simple way to interact with the algorithm. Firstly , the user must choose the camera from which the application receives the video streaming; it can be a local camera or an IP camera. Once the camera, is selected the user can start running the demonstrator. To initialize the algorithm, the user must select the bounding box that defines the target with the mouse.

Through the configuration button, the user can change the parameters of the tracker, as well as different display options.

Figure 3.4: Demonstrator application

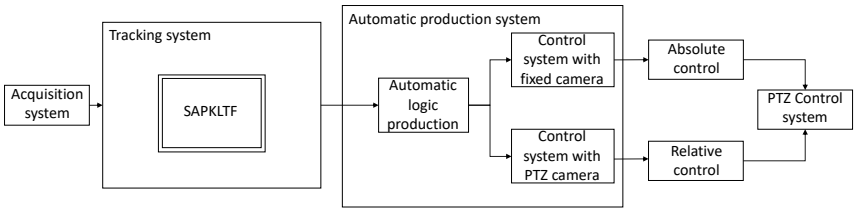


Figure 3.5: PTZ Controller Architecture

3.5.2. PTZ Controller

For the PTZ controller application, another work done at the VPULab [32] is used as starting point: the PTZ control rules of the original work [1] were updated and a people detection module was introduced for automatic initialization of the tracker.

In this work, we have updated the tracker that is used, changing the original PKLTF that was previously used by the new version, SAPKLTF, proposed in this work.

Chapter 4

Evaluation

4.1. Datasets

4.1.1. VOT 2016

The VOT 2016 [6] dataset has 60 sequences. Figure 4.1 shows selected examples of VOT sequences, one of each annotated with visual attributes: camera motion, size change, illumination change, occlusion, and motion change. Rotated boxes can be seen in Figure 4.1. The ground-truth bounding box has the following format the x and y coordinates of each vertex of the rotated rectangle.

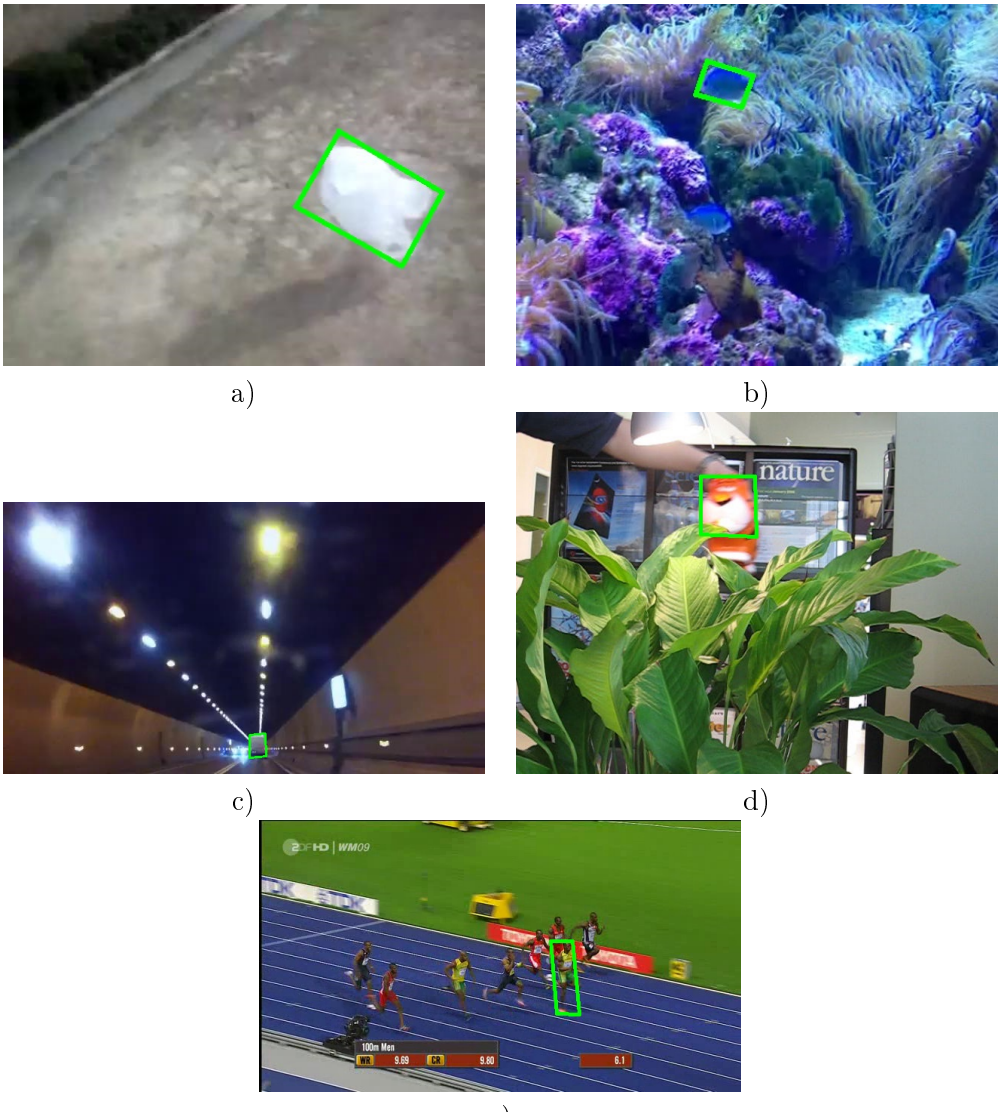
4.1.2. LTDTL 2014

The LTDT [4] is a dataset with 6 sequences (see Figure 4.2). In this case, the bounding box is a rectangle without rotation. The format of this ground-truth is the following: x and y coordinates of the left upper corner, and width and height of the rectangle. If the target is out of scene, or the occlusion is bigger than the 50%, each field is annotated with NaN values.

4.1.3. TTds

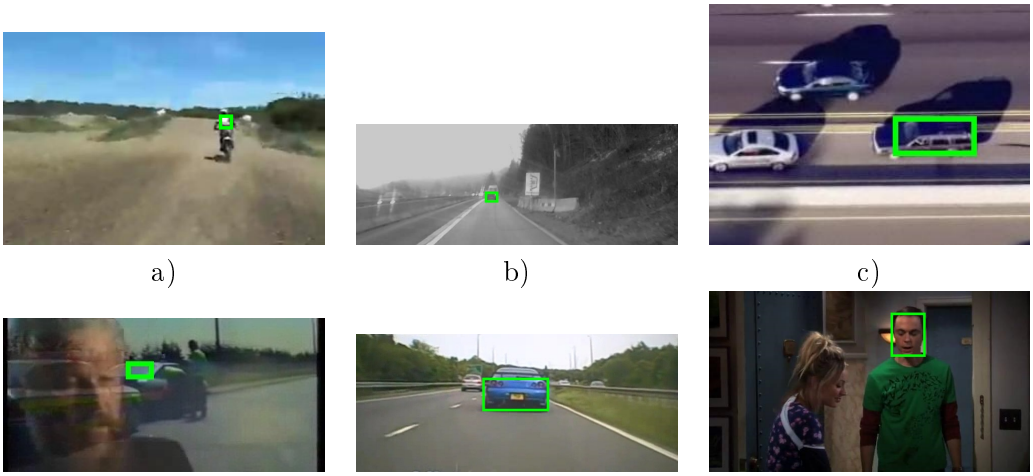
The TTds [1] dataset has 12 sequences (see Figure 4.3). This dataset did not have ground-truth. In this work, the ground-truth is built with a per-frame annotation of the rectangle that contains the target. The annotation is performed with Video Image Annotation Tool[1]. The bounding box annotated is a rectangle without rotation, the format of these ground-truth is the following: x and y coordinates of left upper corner,

[1] Video Image Annotation Tool. https://sourceforge.net/projects/via-tool/



a) b) c) d) e)

Figure 4.1: VOT 2016 example sequences of each annotated visual attributes: a) camera motion; b) size change; c) illumination change; d) occlusion; and e) motion change



a) b) c)

d) e) f)

Figure 4.2: LTDT 2014 sequences. a) 07_motocross; b) 08_volkswagen; c) 09_carchase; d) LiveRunCropped; e) NissanSkylineChaseCropped; and f) Sitcom

and width and height of the rectangle. If the target is out of scene, the annotation is NaN values for each field.

4.2. Comparative Evaluation

In this section, the comparative evaluation between the PKLT; the proposed enhancements: the PKLTF with the CBWH method implemented (BASECBWH) and the PKLTF with the CBWH and the spatial filtering (BASECBWHOUT); and the final SAPKLTF are presented over the selected datasets. Finally, the comparative evaluation of SAPKLTF against State-of-the-Art trackers is presented, showing that the SAPKLT tracker improves the performance of the previous tracker and outperforms State-of-the-Art trackers in determined situations.

4.2.1. VOT 2016

The SAPKLTF algorithm improves the performance of the original PKLTF (see Table 4.1). The PKLTF with CBWH enhances the original algorithm in performance. The spatial filtering incorporated to the CBWH provides a slight enhancement, but the improvement is remarkable only in execution time. In the SAPKLT, the main improvement is produced in Spatial Overlap, mainly due to the introduction of the scale adaptation; also the center error is reduced by the BRW method, although the CBWH method offers better enhancement.

In the comparative with the State-of-the-Art (see Table 4.2),m the SAPKLTF is

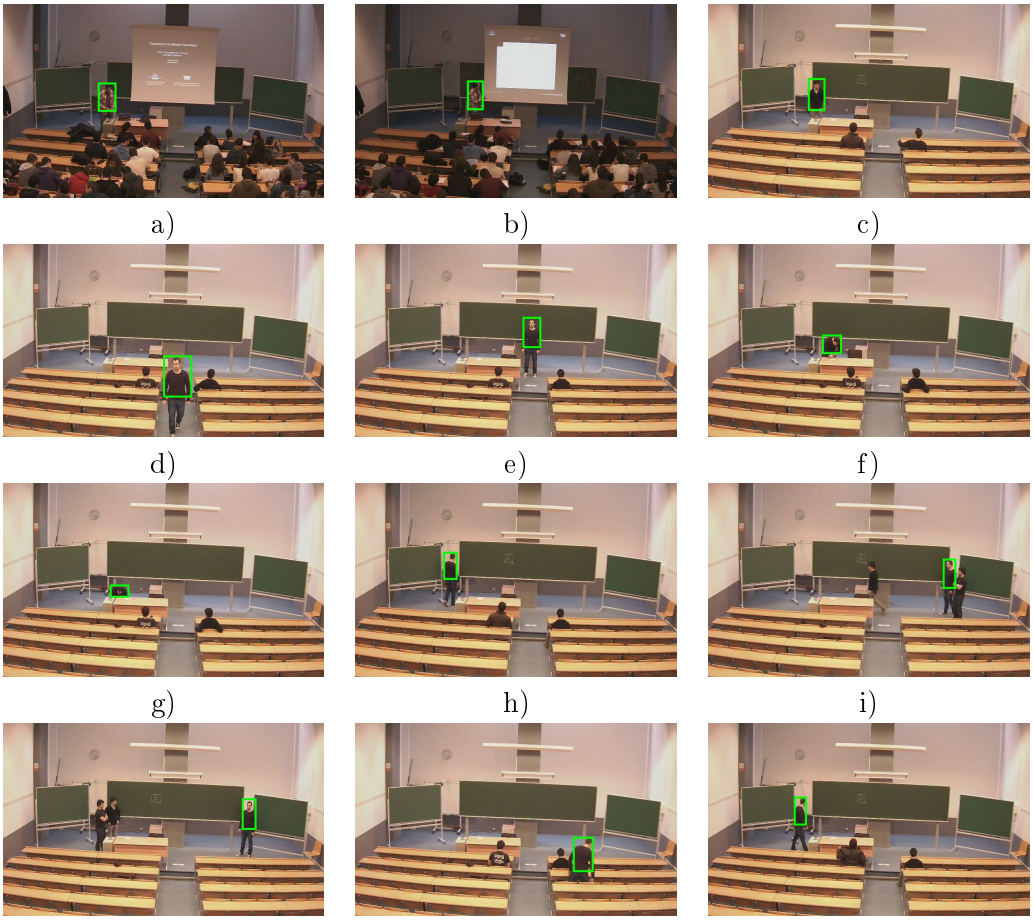


Figure 4.3: TTds sequences. a) L1; b) L2; c) C1; d) C2; e) C3; f) C4; g) C5; h) C6; i) C7; j) C8; k) C9 and l) C10

	SO (%)	Diff. (%)	CE (pixels)	Diff. (%)	Speed (ms)	Diff. (%)
PKLTF	33.16	-	106.1	-	52.47	-
+CBWH	*35.73*	7.19	**96.52**	9.93	*44.77*	17.20
+Spatial Filtering	34.15	2.89	100.9	5.15	**37.83**	38.70
SAPKLTF	**41.63**	20.35	*98.32*	7.91	47.66	10.09

Table 4.1: VOT 2016 Improvements Comparison. The best in bold and the second in cursive letters.

	SO (%)	Diff. (%)	CE (pixels)	Diff. (%)	Speed (ms)	Diff. (%)
C-COT	*55.64*	-14.18	*59.99*	-19.74	2784	-99.18
MDNET	**64.83**	-	**48.14**	-	1873	-98.78
ASMS	41.93	-35.32	95.93	-49.81	**22.95**	-
TLD	52.25	-19.41	64.47	-25.31	190.3	-87.94
LT-FLO	24.65	-61.97	124.40	-62.29	1143	-97.99
PKLTF	33.16	-48.85	106.10	-54.62	52.20	-56.03
SAPKLTF	41.63	-35.79	98.32	-51.03	*47.66*	-51.85

Table 4.2: VOT 2016 Comparison. The best in bold and the second in cursive letters.

close in the mean of Spatial Overlap to the ASMS, but in the median value of Spatial Overlap the SAPKLTF outperforms the ASMS (see Figure B.13). The execution performance of SAPKLTF is better than the original PKLTF. In general, the SAPKLTF tracker obtains a good performance in , the VOT 2016 dataset. Detailed results can be found in Appendix B.1.

4.2.2. LTDT 2014

In the LTDT 2014 dataset, the performance of SAPKLTF improves the original PKLTF performance (see Table 4.3). In this case, the PKLTF with CBWH improves the performance, but only the execution time enhancement is remarkable. Furthermore, the incorporation of the spatial filtering provides a notable enhancement; the performance in Center Error is the best of the proposed methods.

The SAPKLTF has an important improvement in performance, with a remarkable enhancement in the Spatial Overlap.

These performance improvements put the SAPKLTF ahead the ASMS and C-COT, and close to the LT-FLO tracker (see Table 4.4). See Appendix B.2 for the detailed results.

	SO (%)	Diff. (%)	CE (pixels)	Diff. (%)	Speed (ms)	Diff. (%)
PKLTF	10.64	-	93.98	-	49.40	-
+CBWH	10.93	2.65	92.65	1.44	**38.64**	27.85
+Spatial Filtering	*14.36*	25.91	**80.36**	16.95	*40.73*	21.29
SAPKLTF	**17.83**	40.33	*85.89*	9.42	43.83	12.71

Table 4.3: LTDT 2014 Improvements Comparison. The best in bold and the second in cursive letters.

	SO (%)	Diff. (%)	CE (pixels)	Diff. (%)	Speed (ms)	Diff. (%)
C-COT	15.71	-69.51	165.6	-81.83	6340	-99.82
MDNET	*44.17*	-14.28	*65.85*	-54.31	1592	-99.30
ASMS	14.12	-72.60	124.4	-75.81	**11.19**	-
TLD	**51.53**	-	**30.09**	-	90.86	-87.68
LT-FLO	23.98	-53.46	81.16	-62.92	502.3	-97.77
PKLTF	10.64	-79.35	93.98	-67.98	49.40	-77.35
SAPKLTF	17.83	-65.40	85.89	-64.97	*43.83*	-74.47

Table 4.4: LTDT 2014 Comparison. The best in bold and the second in cursive letters.

4.2.3. TTds

The PKLTF has a good performance in the TTds dataset because this tracker was designed specifically for this kind of situations and it is optimized for that. Table 4.5 shows that the CBWH and the spatial filtering do not provide any improvement in Spatial Overlap nor in Center Error, only the execution time is better in these approaches. Nevertheless, the SAPKLTF slightly enhances the performance of the original algorithm.

In the comparison with the State-of-the-Art trackers (see Table 4.6) , the SAPKLT performs at the level of the top performance trackers; in the average values the performance is slightly below that of ASMS, but in the median value it is slightly higher (see Figure B.21). See Appendix B.3 for the detailed results.

4.3. Conclusion

The performance of the SAPKLTF tracker improves, in general, the performance of the original PKLTF. The SAPKLT performs better in short-term situation after the changes, and preserve the good performance in long-term scenarios.

At individual sequence level (see Appendix B), in terms of Spatial Overlap, the

	SO (%)	Diff. (%)	CE (pixels)	Diff. (%)	Speed (ms)	Diff. (%)
PKLTF	*53.13*	-	*62.52*	-	41.24	-
+CBWH	50.04	-6.17	73.71	-15.18	*28.31*	45.67
+Spatial Filtering	47.30	-12.32	89.36	-30.04	**27.45**	50.24
SAPKLTF	**53.31**	0.34	**60.29**	3.70	35.71	15.49

Table 4.5: TTds Improvements Comparison. The best in bold and the second in cursive letters.

	SO (%)	Diff. (%)	CE (pixels)	Diff. (%)	Speed (ms)	Diff. (%)
C-COT	**54.96**	-	77.54	-23.39	7178	-99.83
MDNET	50.48	-8.87	92.76	-35.96	1323	-99.10
ASMS	*53.78*	-2.19	**59.4**	-	**11.86**	-
TLD	47.73	-15.15	86.57	-31.39	215.90	-94.51
LT-FLO	29.46	-86.56	95.67	-37.91	1910	-99.37
PKLTF	53.13	-3.44	62.52	-4.99	41.24	-71.24
SAPKLTF	53.31	-3.09	*60.29*	-1.48	*35.71*	-66.78

Table 4.6: TTds Comparison. The best in bold and the second in cursive letters.

SAPKLTF is the top performer in 6 sequences of VOT 2016 and in 2 sequences of TTds. In addition, in CE measures the SAPKLTF is the best in 10 sequences of VOT 2016 and in 3 sequences of TTds. Also, the proposed approach can work in real-time constraints.

Chapter 5

Conclusions and future work

5.1. Conclusions

First of all, an extensive analysis concerning the related work was done. It was required to understand the video object tracking process, likewise to study existing tracking approaches, as well as, metrics and datasets used for video tracking evaluation.

Following this in-depth analysis, different improvements were proposed. These improvements were implemented over the base algorithm, PKLTF (Point-based Kanade Lucas Tomasi colour-Filter). After that, an improvement version of the approach, SAPKLTF (Scale Adaptive Point-based Kanade Lucas Tomasi colour-Filter), was presented. Once, the SAPKLTF was developed, this was integrated into two applications; a demonstrator and PTZ controller.

Finally, the performance of SAPKLTF was evaluated. For that purpose, a comparative evaluation with a selection of trackers and datasets was performed.

5.2. Future work

Results obtained in the work prove that the proposed algorithm performs better than the base algorithm. However, it is important to notice that those results could be improved since there are some limitations in different scenarios.

We identify some main areas for future work:

- Automatic initialization of the target: the study of different initialization approaches based on object detection, saliency, or objectness proposals are proposed.

- Re-identification improvements: currently after loosing the target, the last model is used for looking for the target object to be re-identified as the target. The feasibility study of an approach that uses multiples models is proposed.
- Model update: currently, the model is re-initialized to perform the updating, other update ways can be studied.
- Adaptation of the tracker to other image domains such as infrared or depth modalities.
- Target model: Introduce more complexity to the model, e.g., introduce information of other color spaces to the histogram.
- Multi-target tracking: Introduce the capability to track multiple target.

.

Bibliography

[1] A. González, R. Martín-Nieto, J. Bescós, and J. M. Martínez, "Single object long-term tracker for smart control of a ptz camera," *in Proc. of the International Conference on Distributed Smart Cameras*, 2014.

[2] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[3] M. Kristan *et al.*, "The visual object tracking vot2014 challenge results," *in Proc. of European Conference on Computer Vision Workshop*, 2014.

[4] O. Camps, R. Cucchiara, A. Del Bimbo, J. Matas, F. Pernici, and S. Sclaroff, "Long-term detection and tracking (LTDT 2014)," *[Online] http://www.micc.unifi.it/LTDT2014/*, 2014.

[5] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

[6] M. Kristan *et al.*, "The visual object tracking vot2015 challenge results," *in Proc. of the IEEE international Conference on Computer Vision workshops*, 2015.

[7] M. Kristan *et al.*, "The visual object tracking vot2016 challenge results," *in Proc. of European Conference on Computer Vision Workshop*, 2016.

[8] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *in Proc. of the 7th International Joint Conference on Artificial Intelligence*, 1981.

[9] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, 2002.

[10] G. Bishop and G. Welch, "An introduction to the kalman filter," *in Proc. of SIGGRAPH, Course*, 2001.

[11] R. Brunelli, *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[12] T. Vojir, J. Noskova, and J. Matas, "Robust scale-adaptive mean-shift for tracking," *Pattern Recognition Letters*, 2014.

[13] G. Zhu, F. Porikli, and H. Li, "Beyond local search: Tracking objects everywhere with instance-specific proposals," *in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[14] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," *in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[15] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," *in Proc. of European Conference on Computer Vision*, 2016.

[16] H. Nam, M. Baek, and B. Han, "Modeling and propagating cnns in a tree structure for visual tracking," *arXiv preprint arXiv:1608.07242*, 2016.

[17] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[18] K. Lebeda, S. Hadfield, J. Matas, and R. Bowden, "Texture-independent long-term tracking using virtual corners," *IEEE Transactions on Image Processing*, 2016.

[19] M. Isard and A. Blake, "Condensation–conditional density propagation for visual tracking," *International Journal of Computer Vision*, 1998.

[20] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," *in Proc. of IEEE Conference Computer Vision and Pattern Recognition*, 2000.

[21] A. Bordes, L. Bottou, P. Gallinari, and J. Weston, "Solving multiclass support vector machines with larank," *in Proc. of the 24th international conference on Machine learning*, 2007.

[22] M. Kristan *et al.*, "The visual object tracking vot2013 challenge results," *in Proc. of the IEEE International Conference on Computer Vision Workshops*, 2013.

[23] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," *in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[24] A. Li, M. Lin, Y. Wu, M.-H. Yang, and S. Yan, "Nus-pro: A new visual tracking challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[25] J. C. SanMiguel, "Framework for evaluation of single-object video trackers," *[Online] https://bitbucket.org/jcsma/tracking-st*, 2017.

[26] R. Martín and J. M. Martinez, "Correlation study of video object trackers evaluation metrics," *IET Electronics Letters*, 2014.

[27] L. Čehovin, A. Leonardis, and M. Kristan, "Visual object tracking performance measures revisited," *IEEE Transactions on Image Processing*, 2016.

[28] J. Shi and C. Tomasi, "Good features to track," *in Proc. of IEEE Conference Computer Vision and Pattern Recognition*, 1994.

[29] J. Ning, L. Zhang, D. Zhang, and C. Wu, "Robust mean-shift tracking with corrected background-weighted histogram," *IET computer vision*, 2012.

[30] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," *in Proc. of IEEE Workshop on Applications of Computer Vision*, 1998.

[31] J. Sanjuan, "Seguimiento de objetos en tiempo real," *Bachelor Thesis, Univ. Autónoma de Madrid*, 2014.

[32] A. Palero, "Automatización de funciones en el seguimiento del profesor para la emisión de clases presenciales," *Master Thesis, Univ. Autónoma de Madrid*, 2016.

Appendix A

State-of-the-Art comparative evaluation per sequence

A.1. VOT 2016

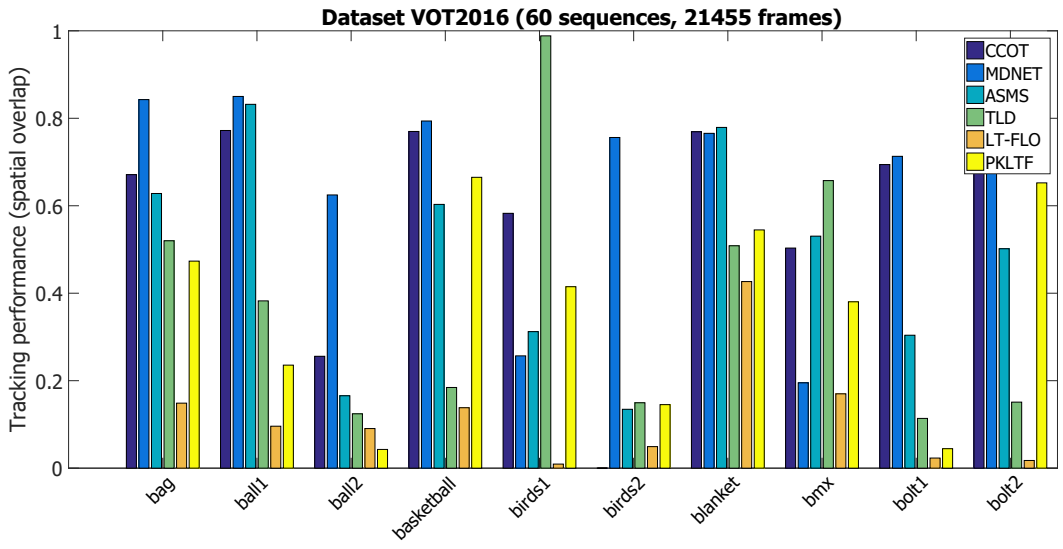


Figure A.1: VOT 2016 Spatial Overlap per sequence (1-10)

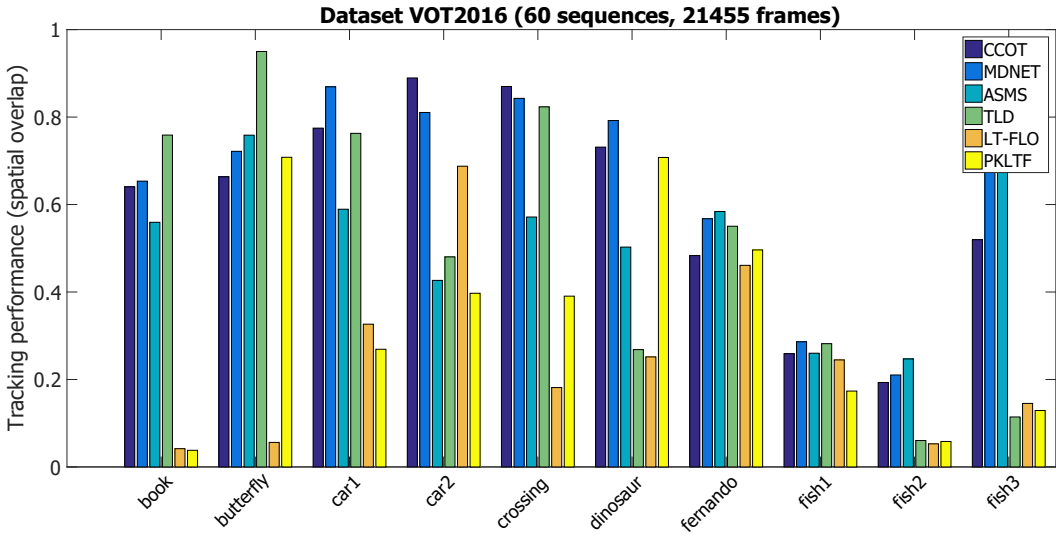


Figure A.2: VOT 2016 Spatial Overlap per sequence (11-20)

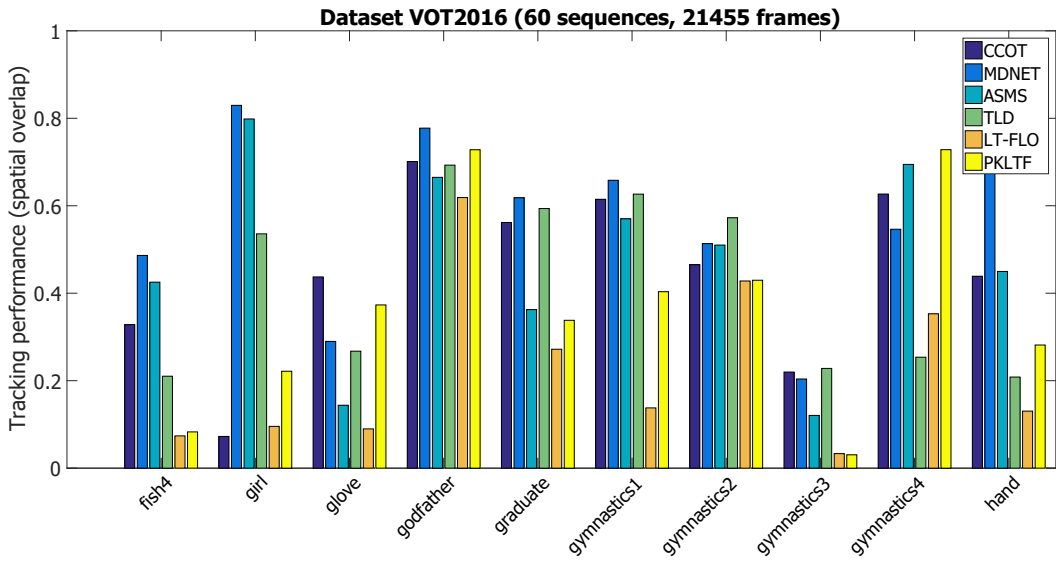


Figure A.3: VOT 2016 Spatial Overlap per sequence (21-30)

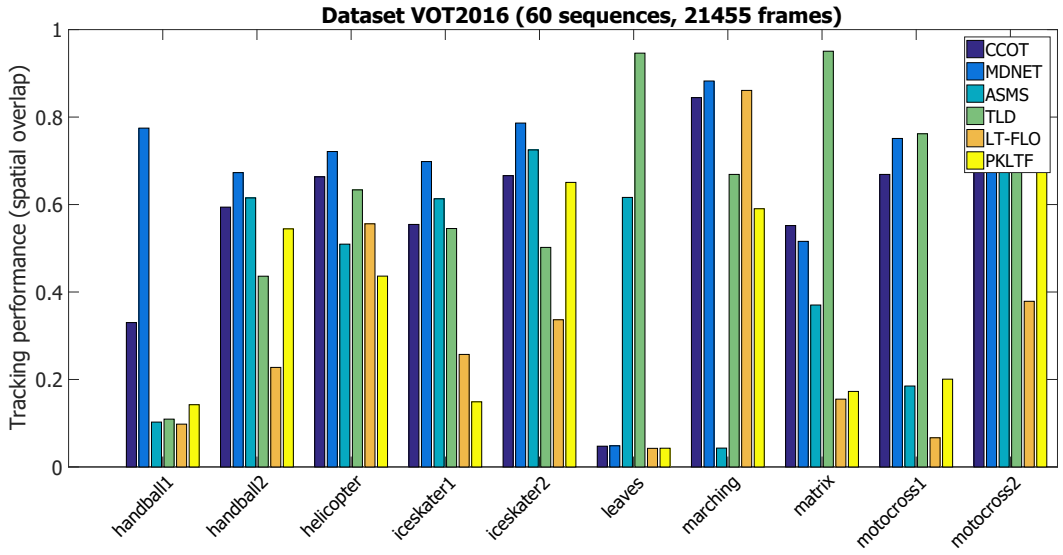


Figure A.4: VOT 2016 Spatial Overlap per sequence (31-40)

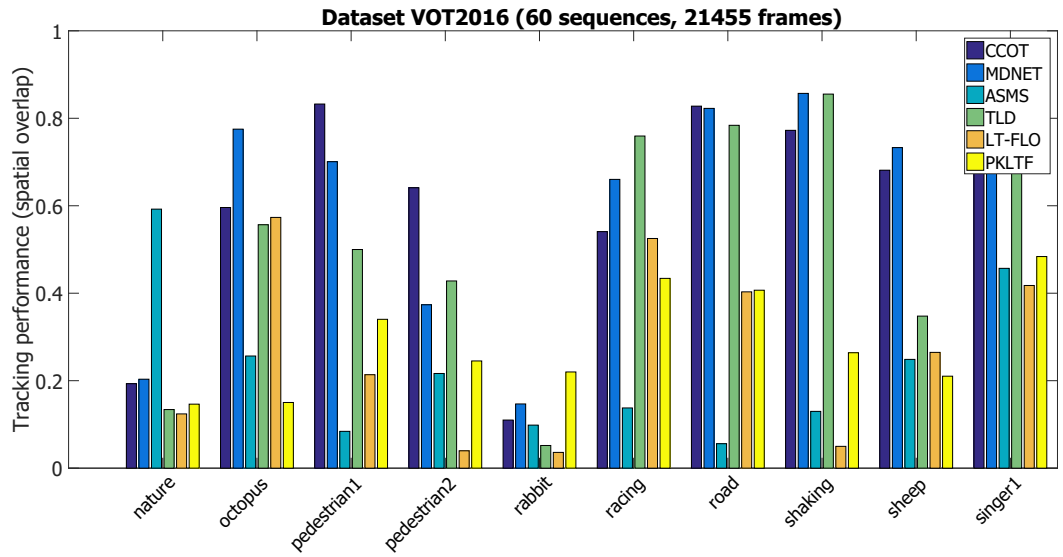


Figure A.5: VOT 2016 Spatial Overlap per sequence (41-50)

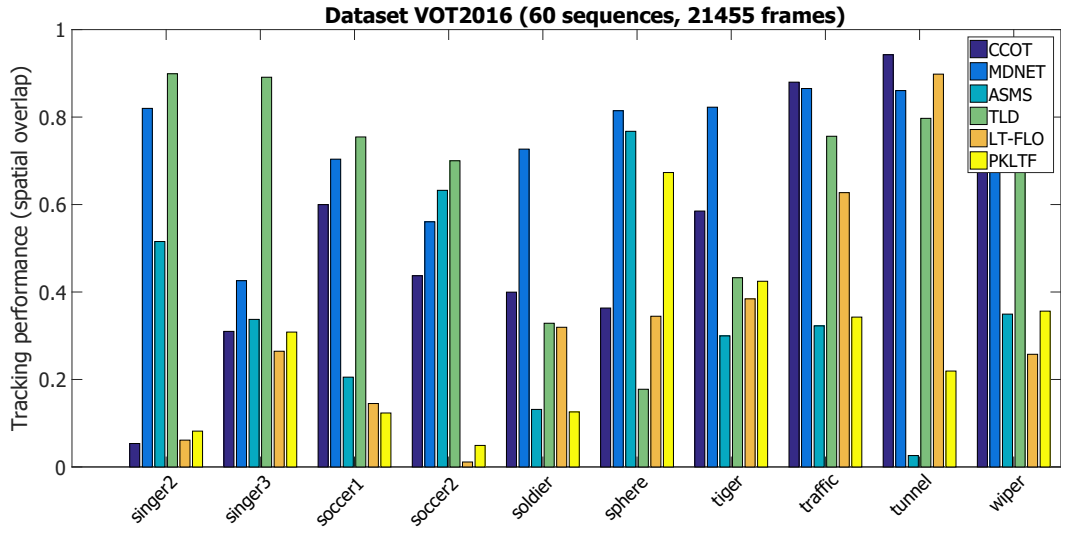


Figure A.6: VOT 2016 Spatial Overlap per sequence (51-60)

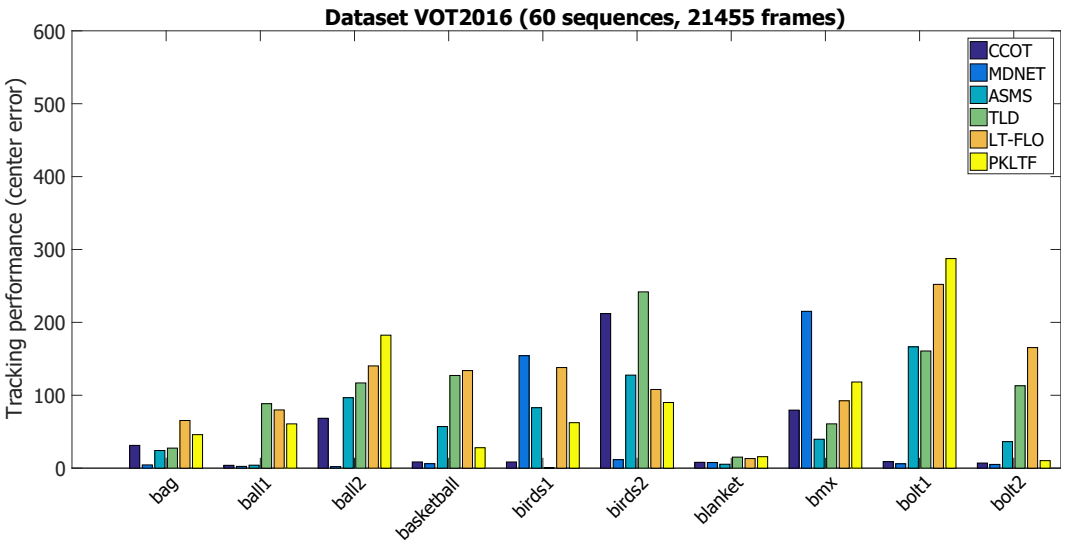


Figure A.7: VOT 2016 Center Error per sequence (1-10)

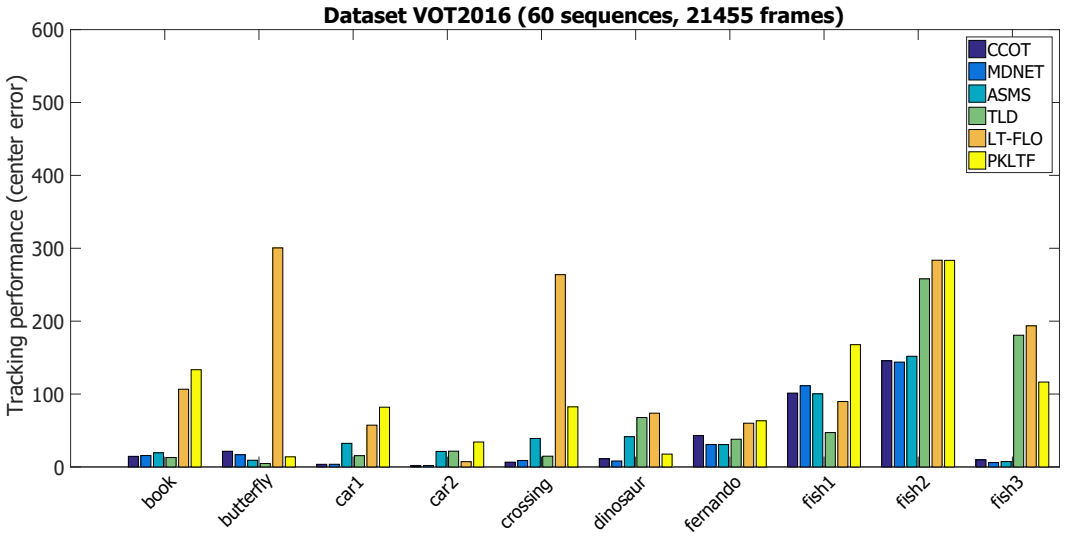


Figure A.8: VOT 2016 Center Error per sequence (11-20)

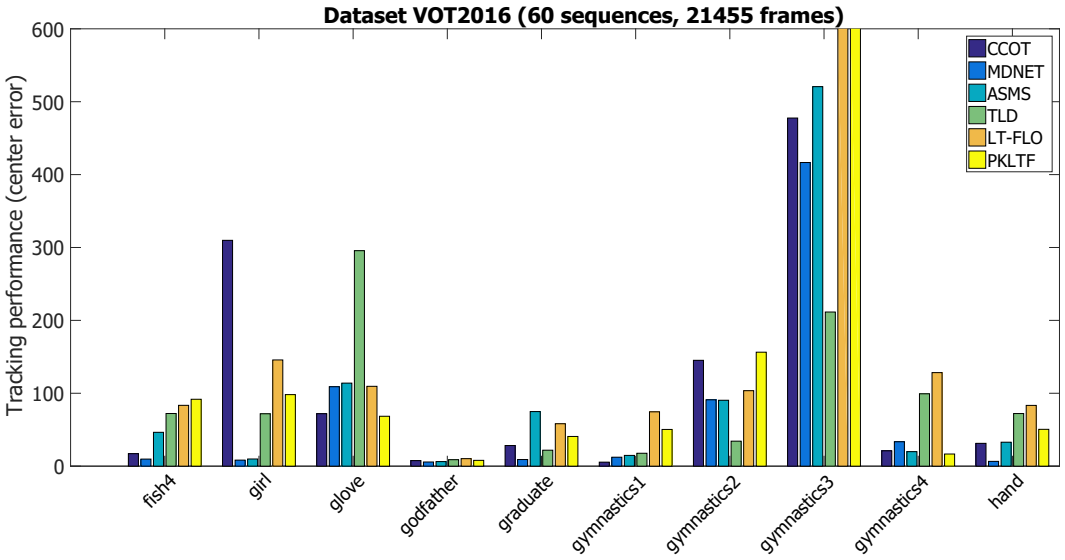


Figure A.9: VOT 2016 Center Error per sequence (21-30)

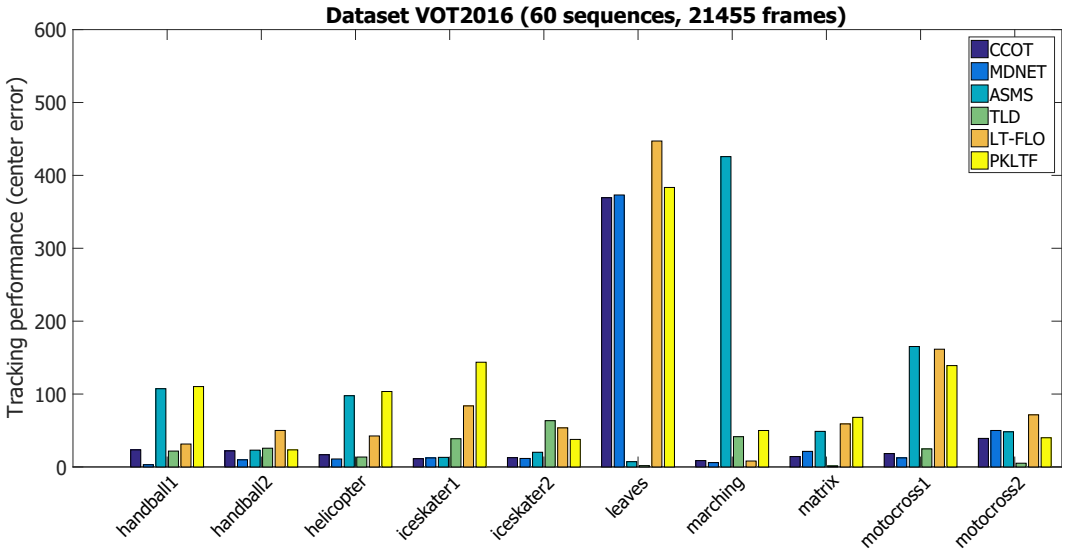


Figure A.10: VOT 2016 Center Error per sequence (31-40)

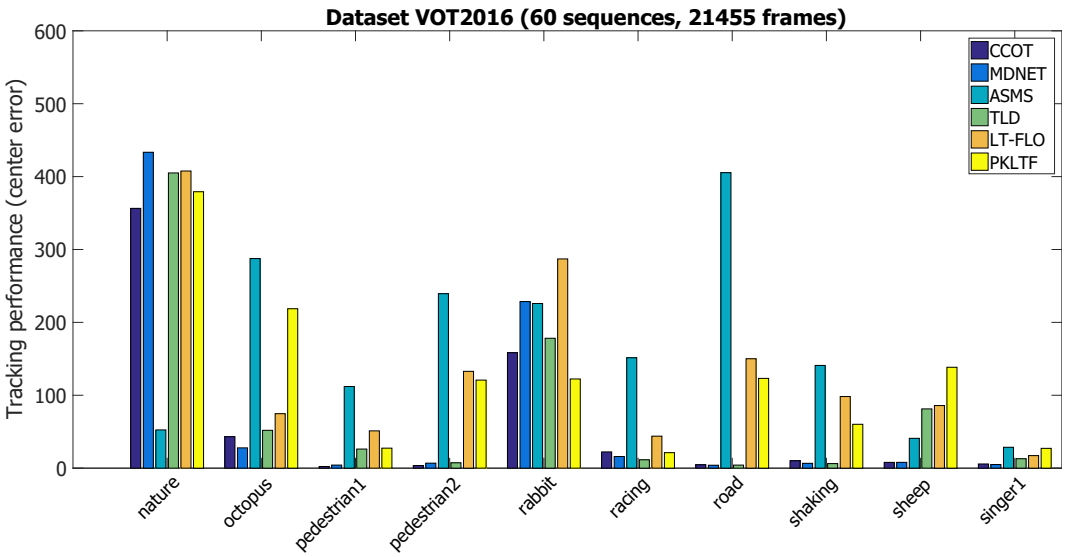


Figure A.11: VOT 2016 Center Error per sequence (41-50)

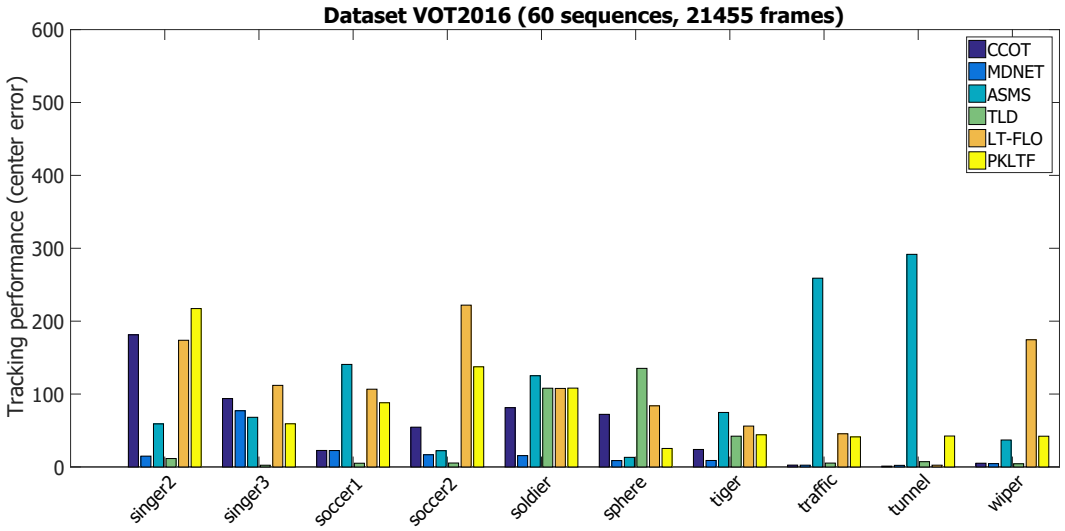


Figure A.12: VOT 2016 Center Error per sequence (51-60)

A.2. LTDT 2014

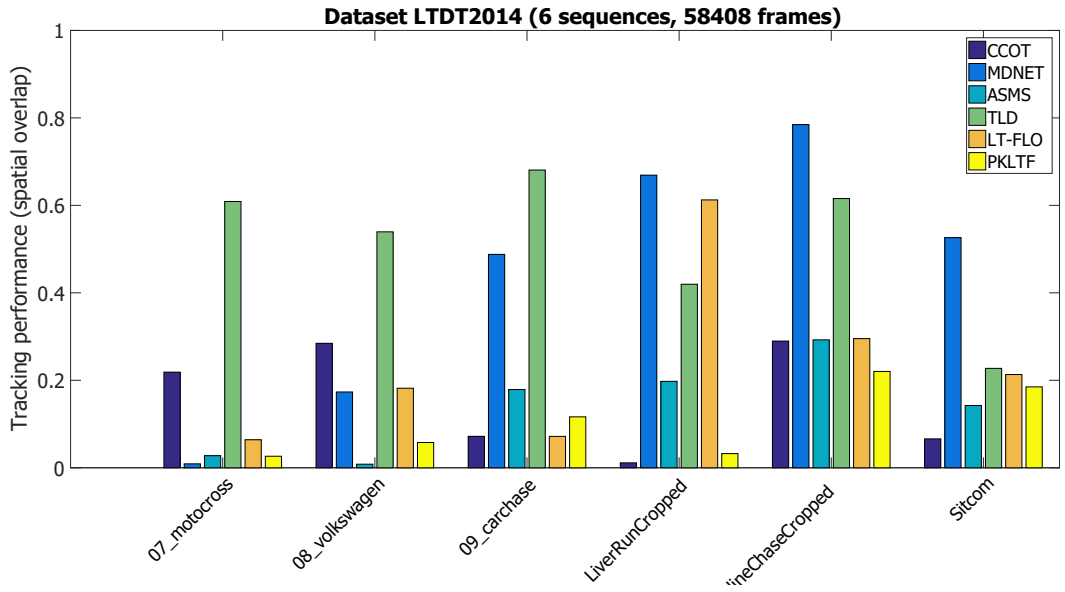


Figure A.13: LTDT 2014 Spatial Overlap per sequence

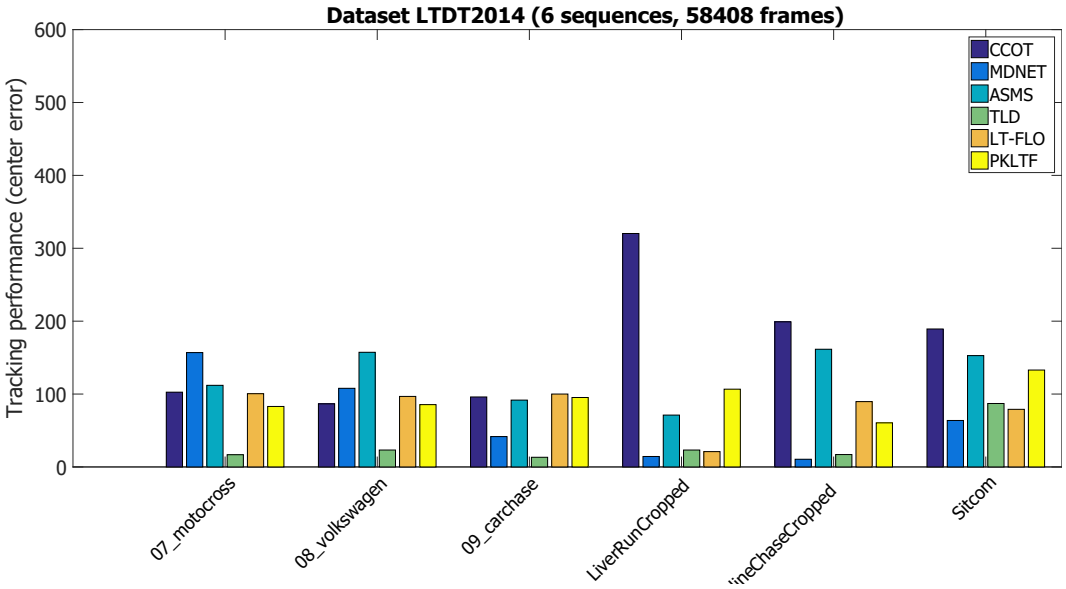


Figure A.14: LTDT 2014 Center Error per sequence

A.3. TTds

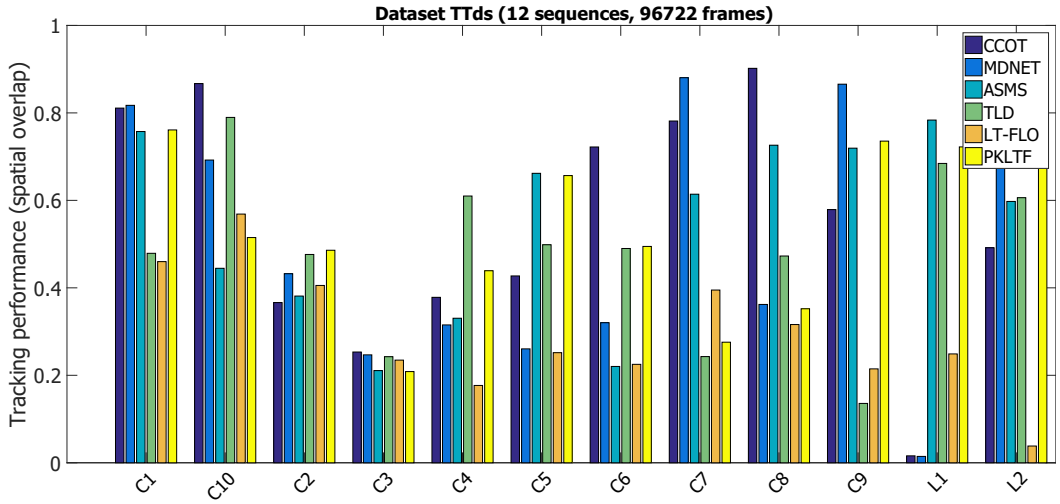


Figure A.15: TTds Spatial Overlap per sequence

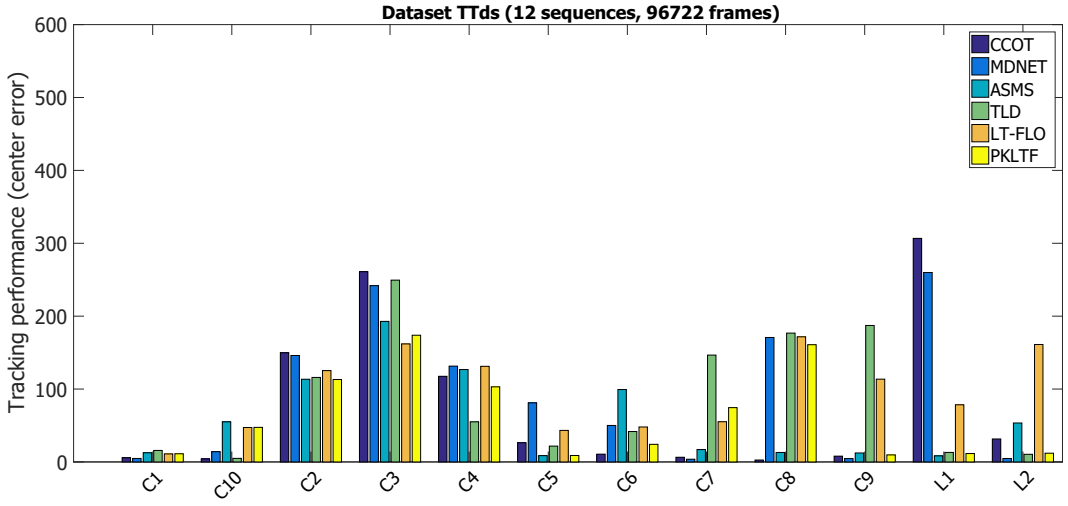


Figure A.16: TTds Center Error per sequence

Appendix B

Results comparison

B.1. VOT 2016

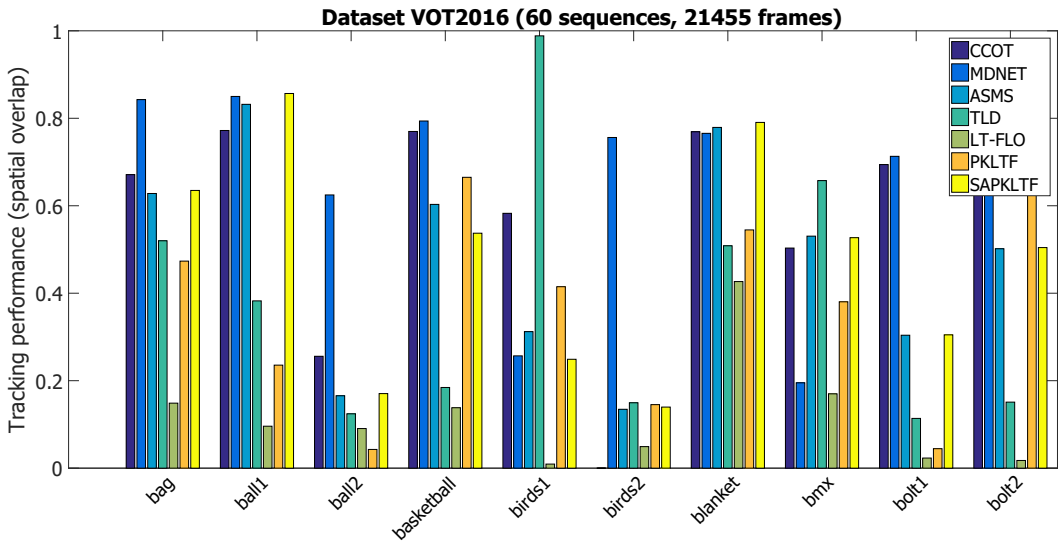


Figure B.1: VOT 2016 Spatial Overlap per sequence (1-10)

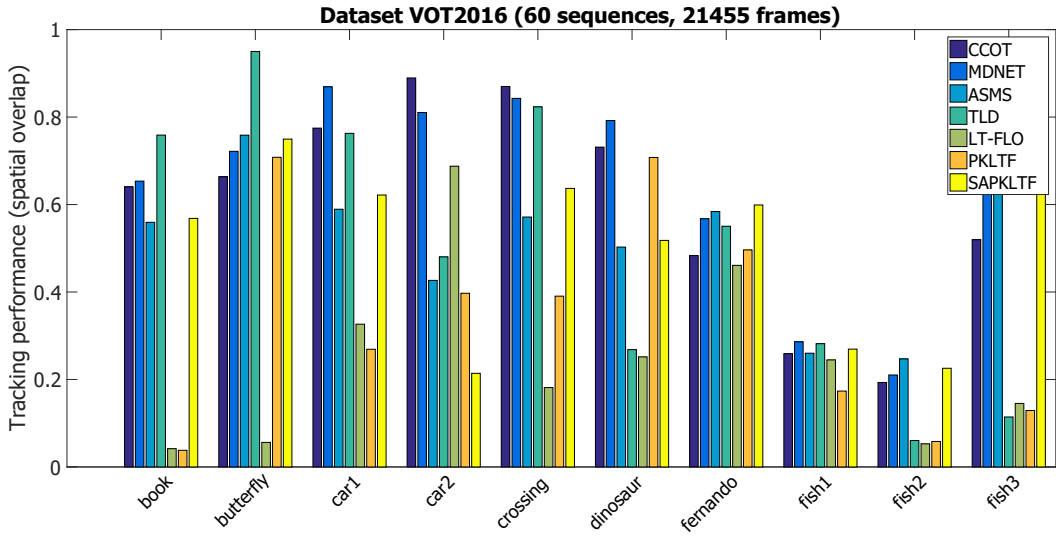


Figure B.2: VOT 2016 Spatial Overlap per sequence (11-20)

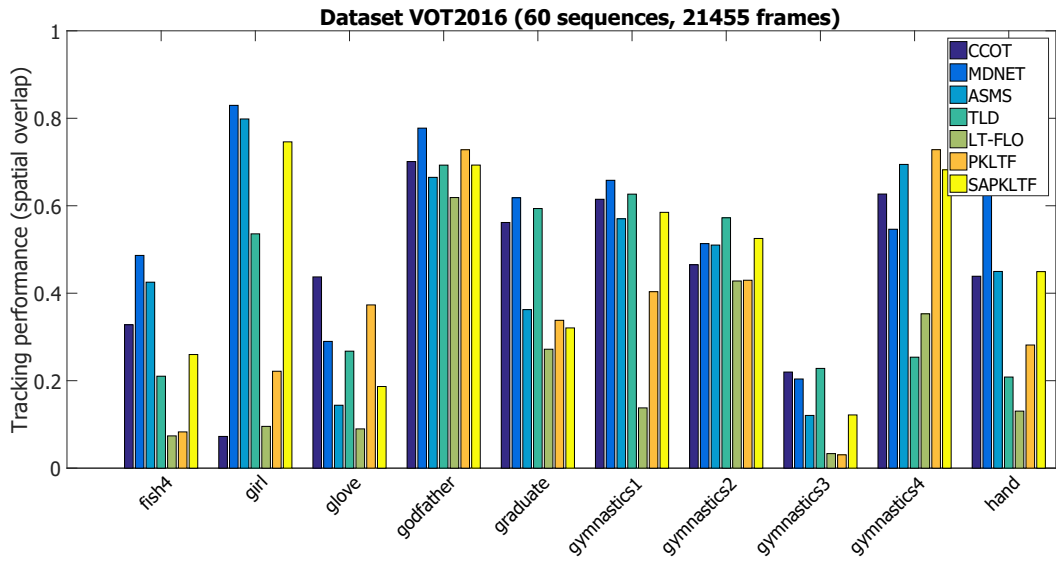


Figure B.3: VOT 2016 Spatial Overlap per sequence (21-30)

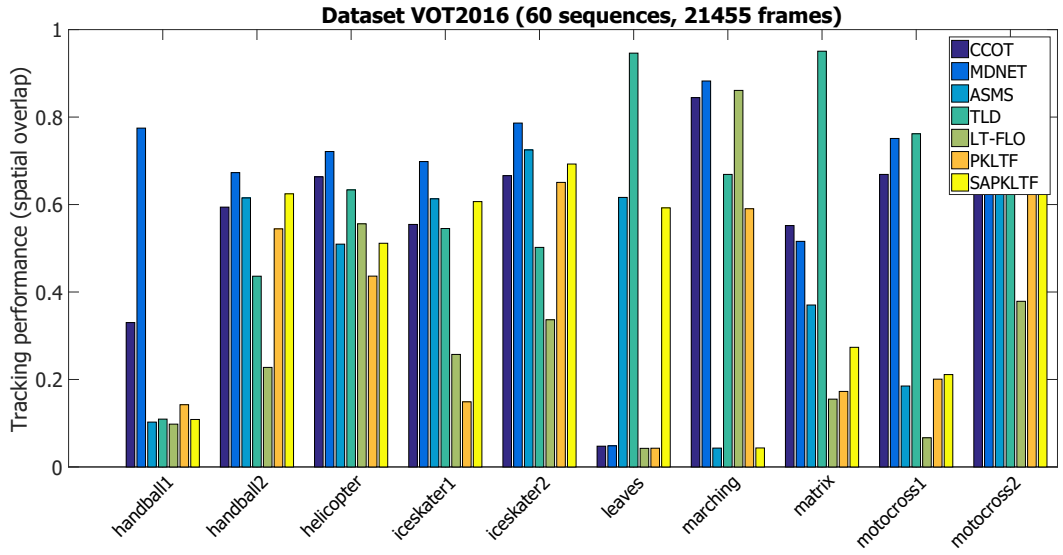


Figure B.4: VOT 2016 Spatial Overlap per sequence (31-40)

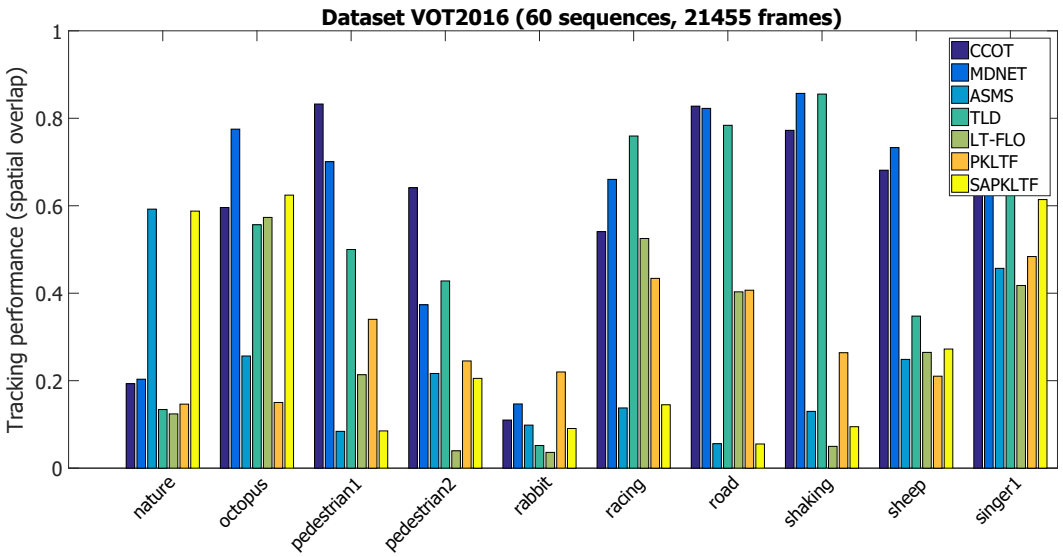


Figure B.5: VOT 2016 Spatial Overlap per sequence (41-50)

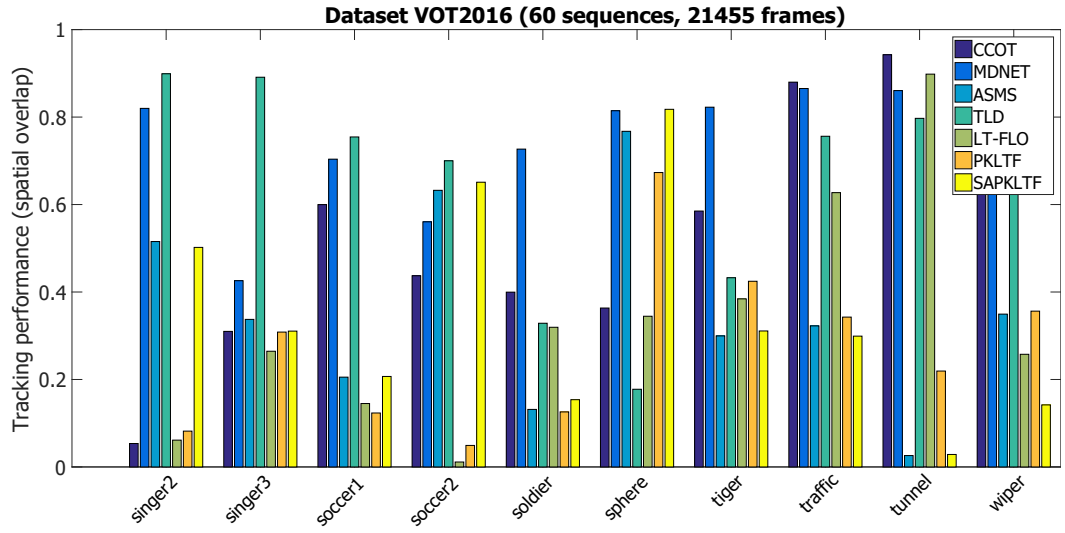


Figure B.6: VOT 2016 Spatial Overlap per sequence (51-60)

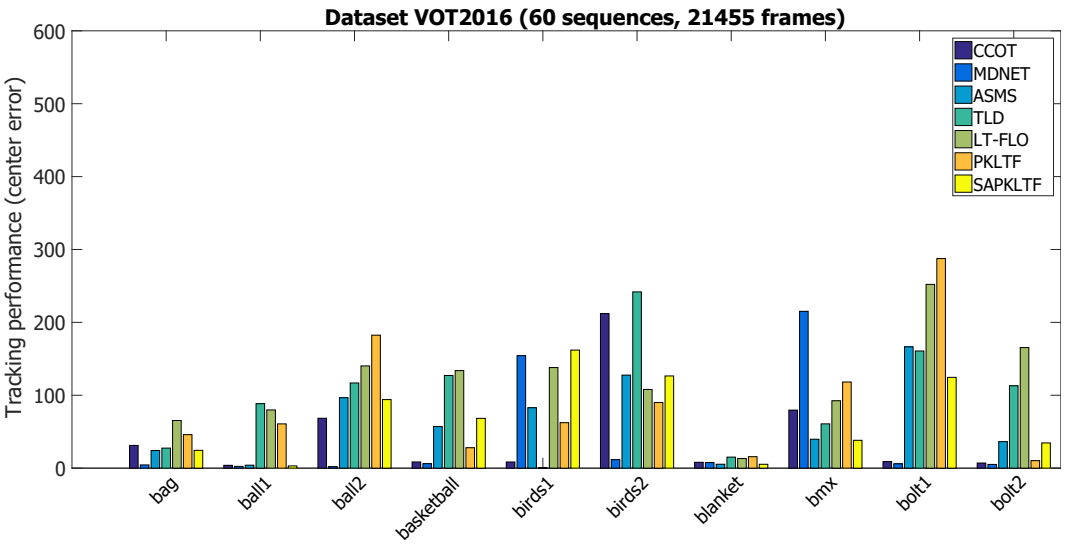


Figure B.7: VOT 2016 Center Error per sequence (1-10)

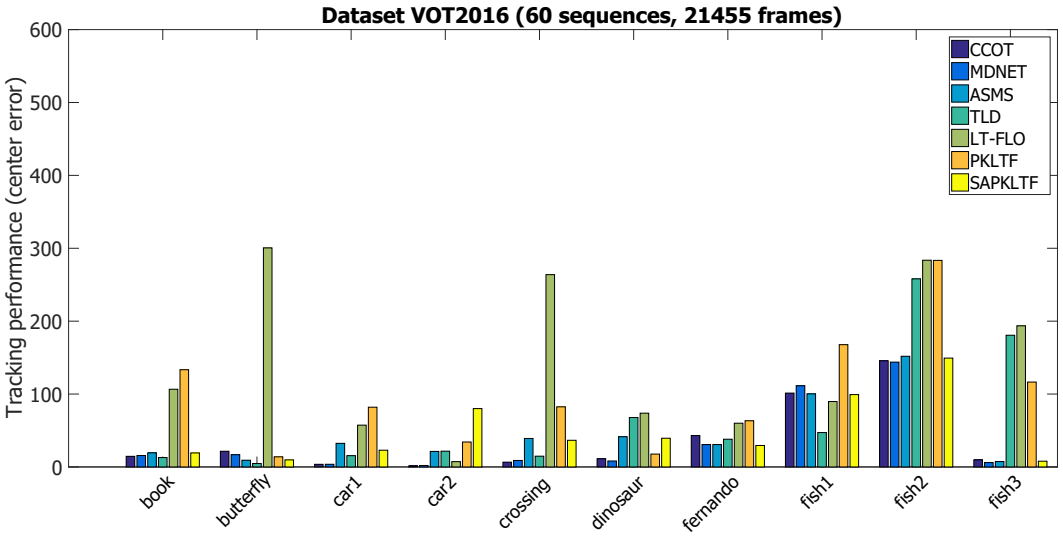


Figure B.8: VOT 2016 Center Error per sequence (11-20)

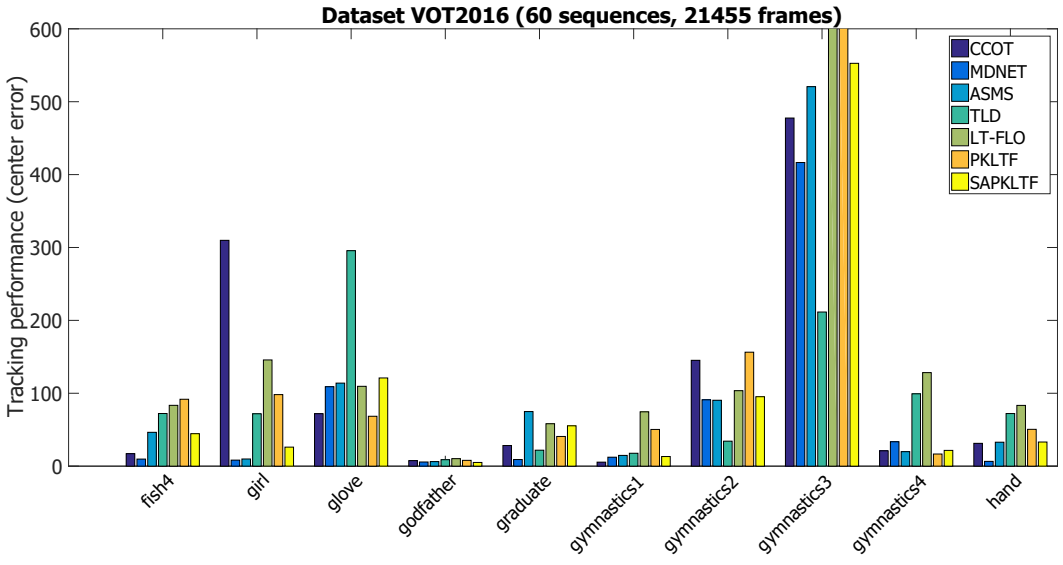


Figure B.9: VOT 2016 Center Error per sequence (21-30)

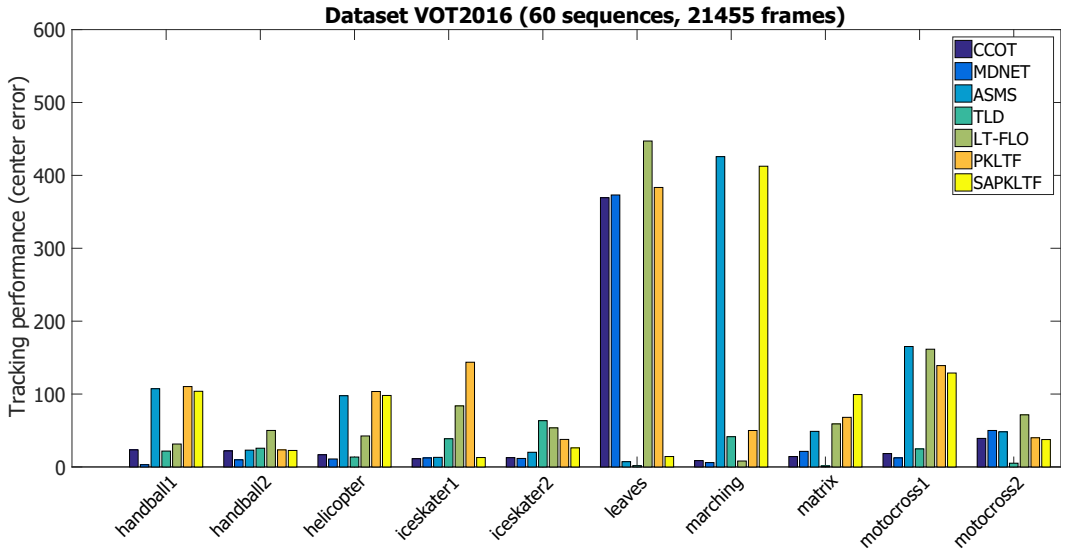


Figure B.10: VOT 2016 Center Error per sequence (31-40)

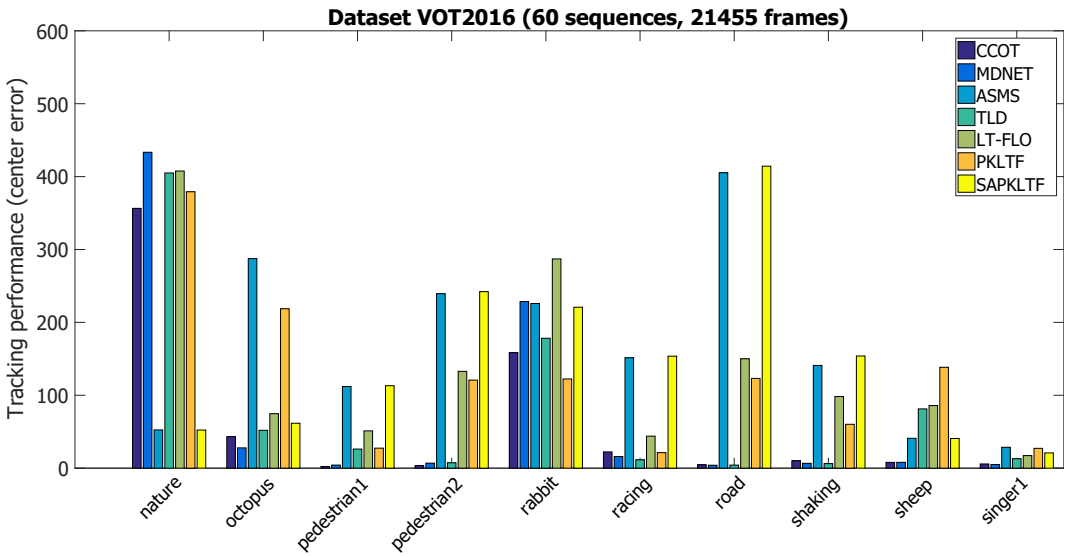


Figure B.11: VOT 2016 Center Error per sequence (41-50)

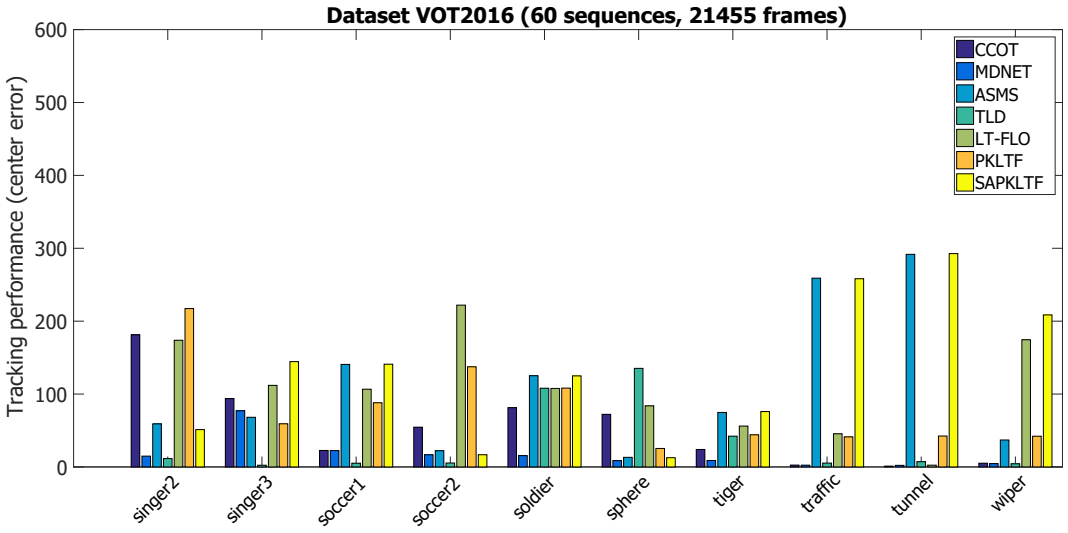
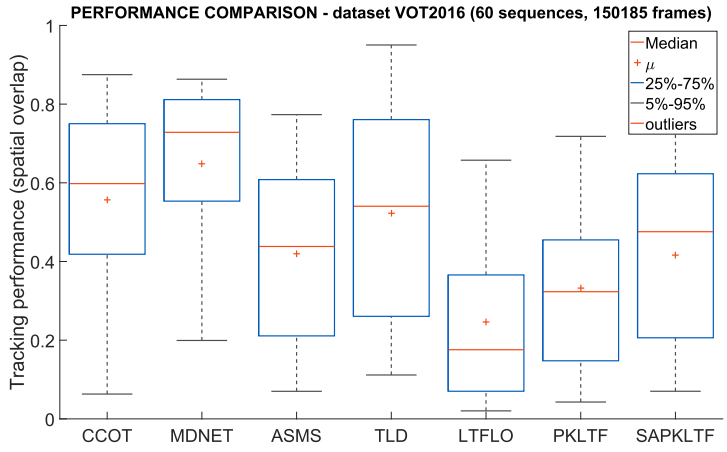
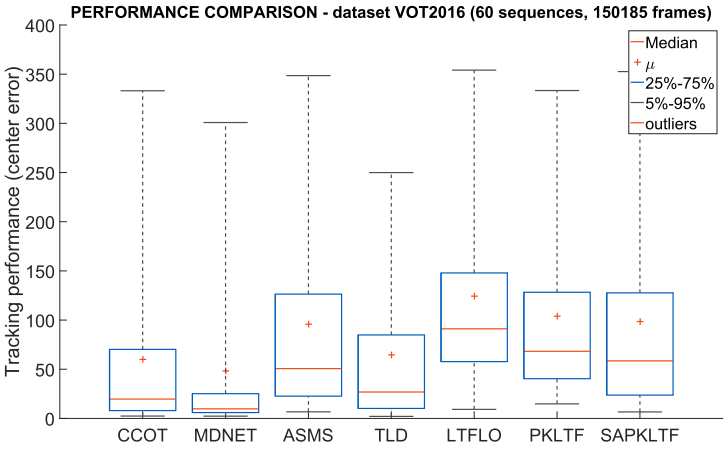


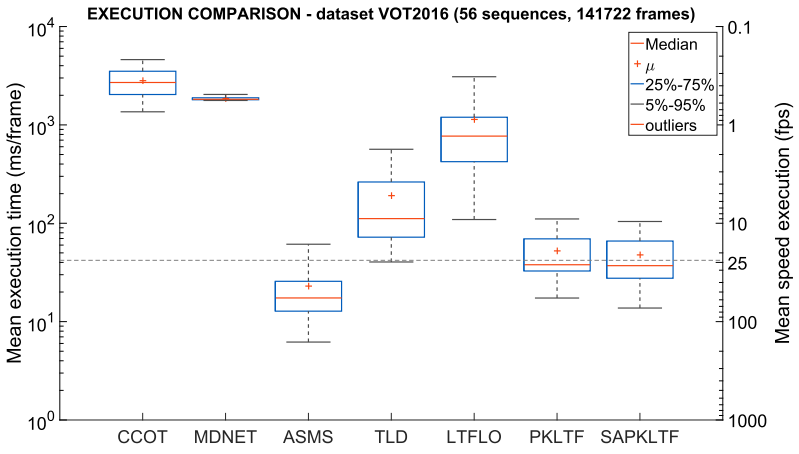
Figure B.12: VOT 2016 Center Error per sequence (51-60)



a)

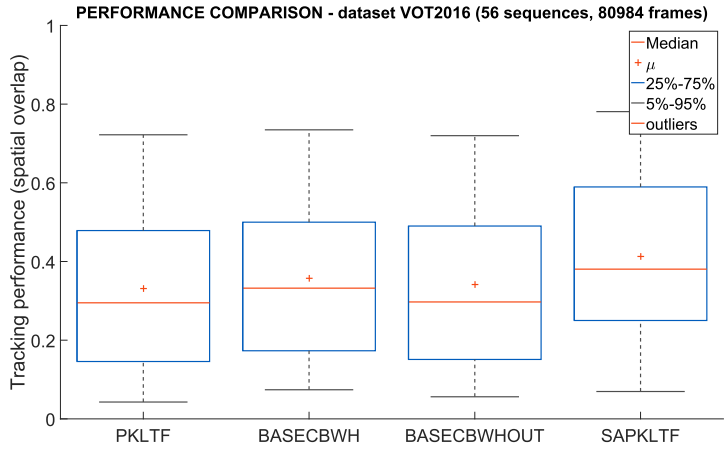


b)

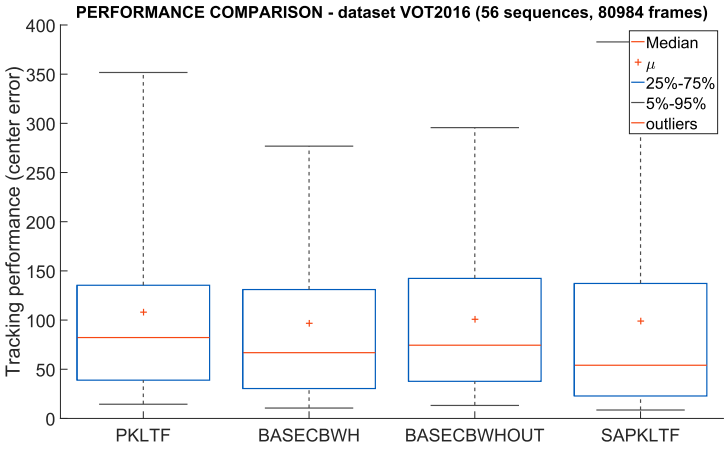


c)

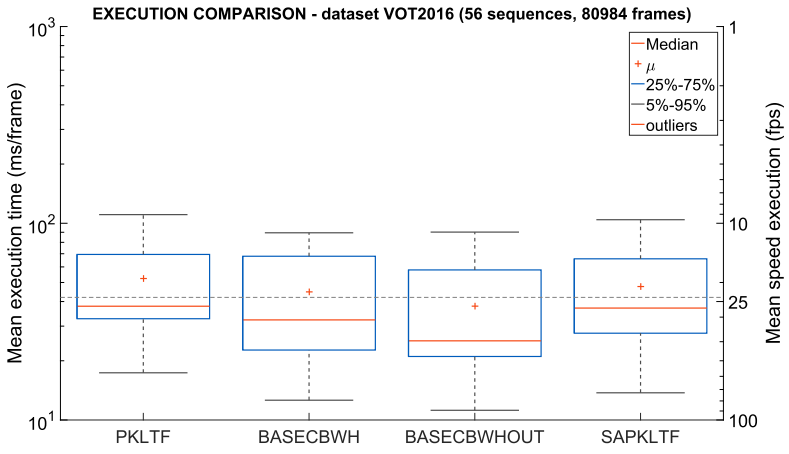
Figure B.13: VOT 2016 Comparison. a) Spatial Overlap; b) Center Error; and c) Execution Time



a)



b)



c)

Figure B.14: VOT 2016 Improvements Comparison. a) Spatial Overlap; b) Center Error; and c) Execution Time

B.2. LTDT 2014

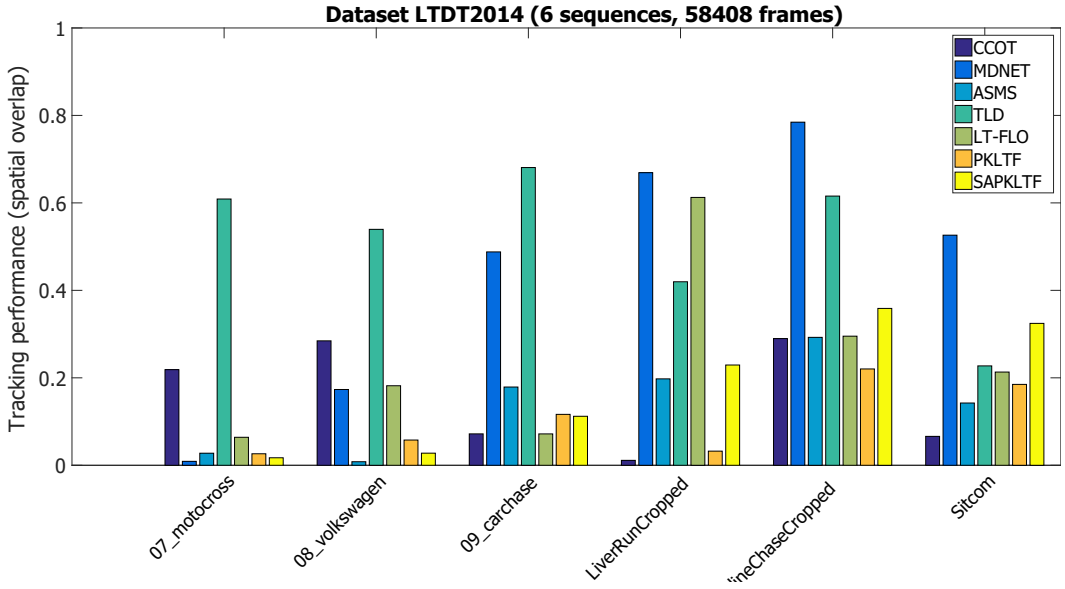


Figure B.15: LTDT 2014 Spatial Overlap per sequence

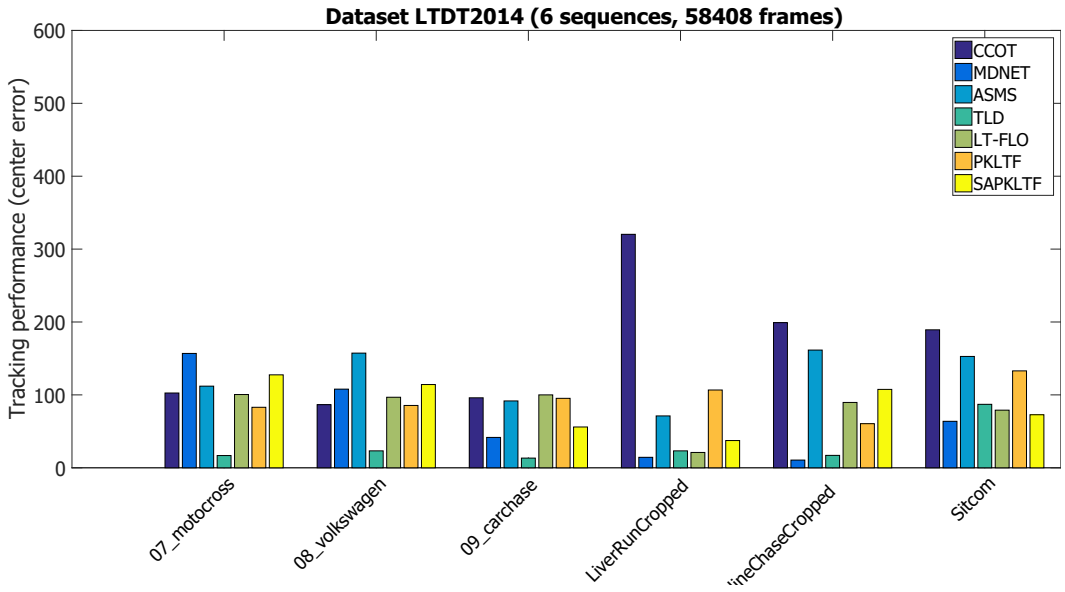
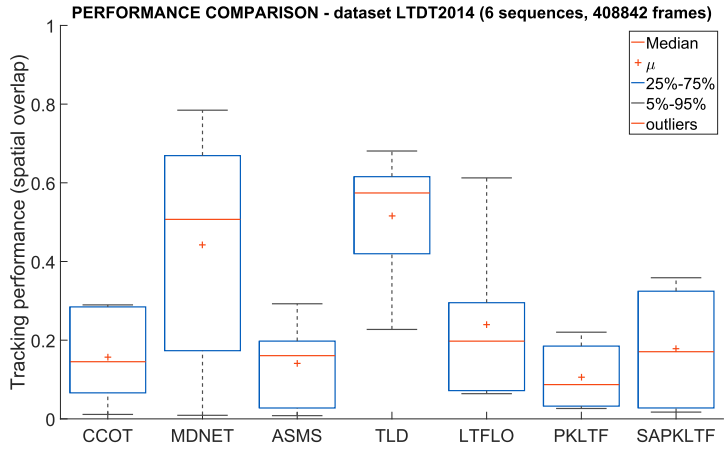
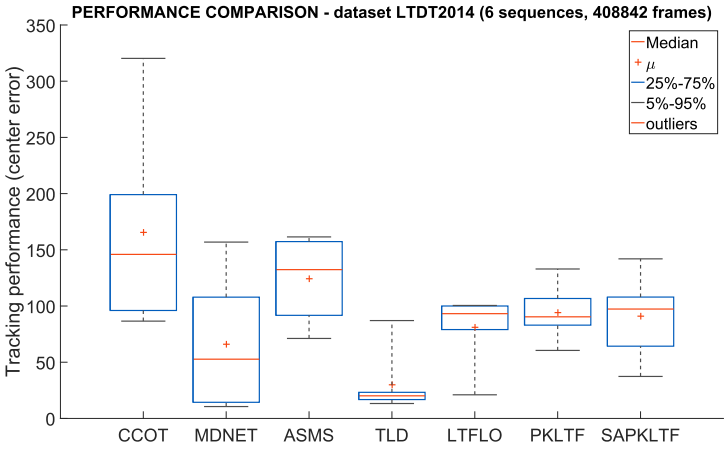


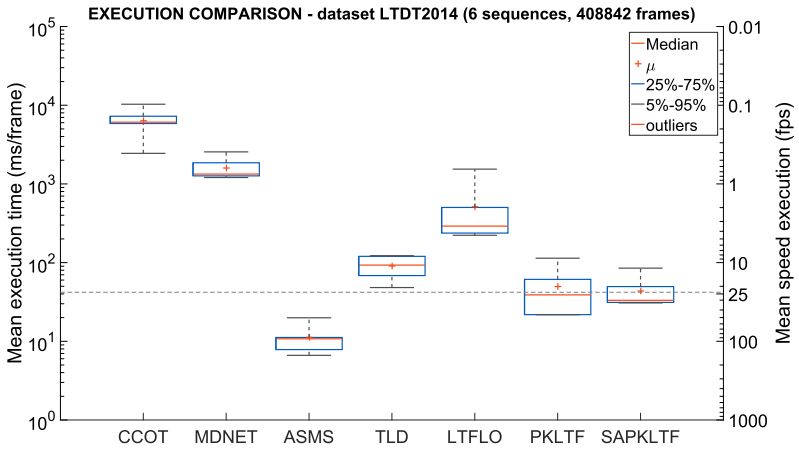
Figure B.16: LTDT 2014 Center Error per sequence



a)

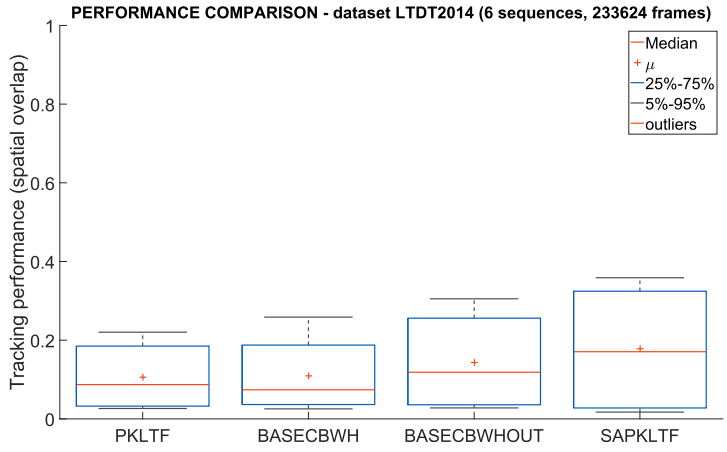


b)

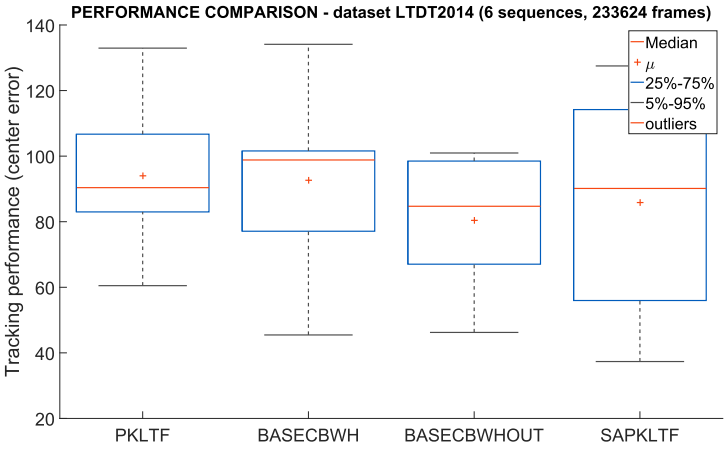


c)

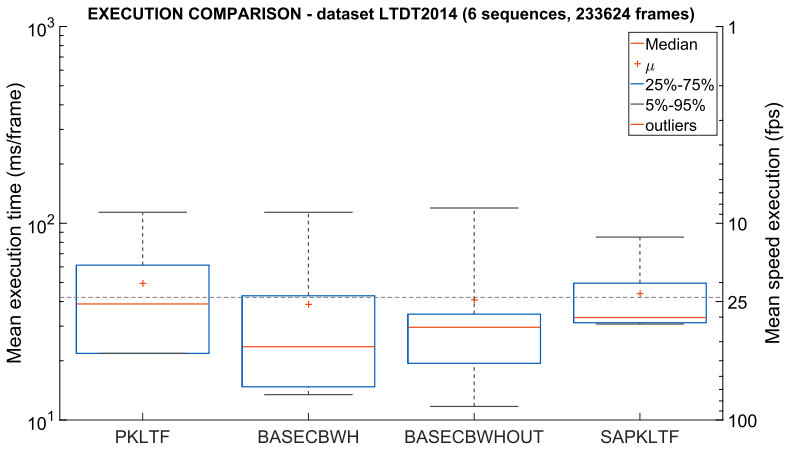
Figure B.17: LTDT 2014 Comparison. a) Spatial Overlap; b) Center Error; and c) Execution Time



a)



b)



c)

Figure B.18: LTDT 2014 Improvements Comparison. a) Spatial Overlap; b) Center Error; and c) Execution Time.

B.3. TTds

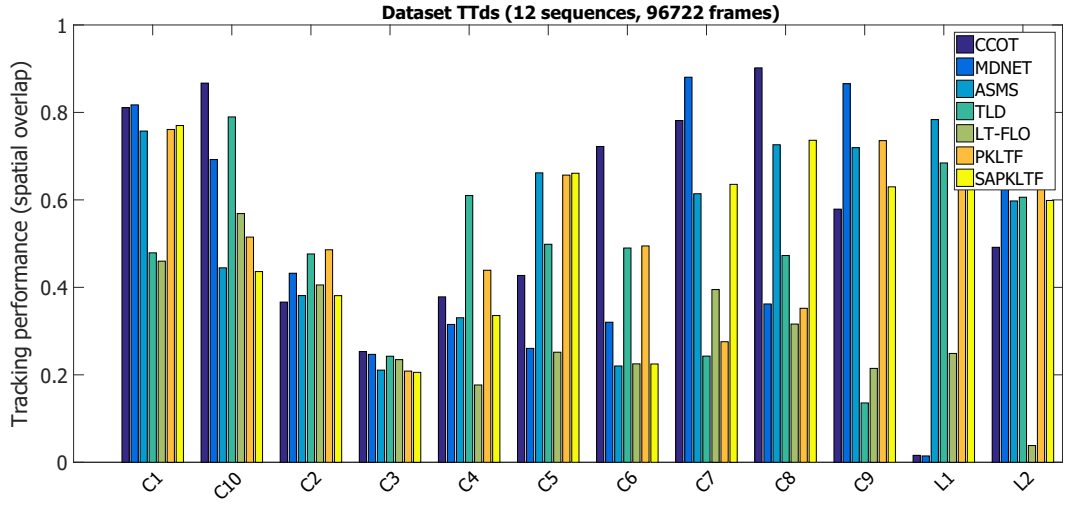


Figure B.19: TTds Spatial Overlap per sequence

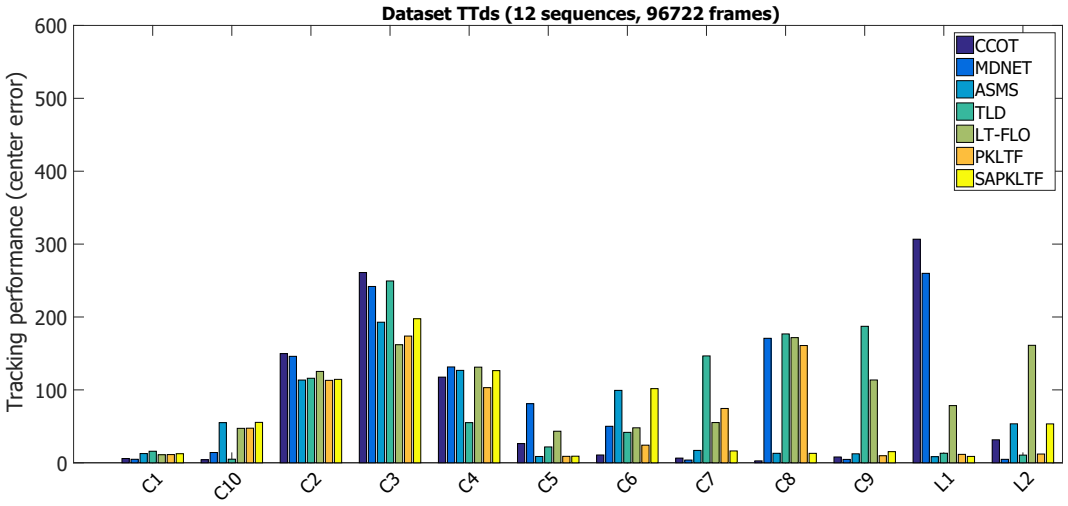
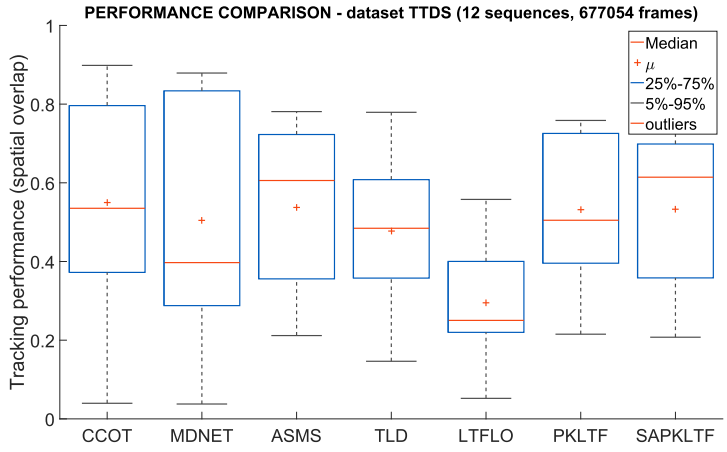
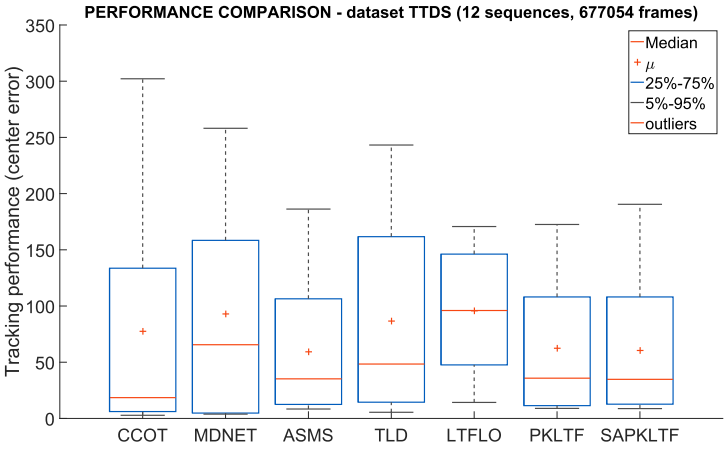


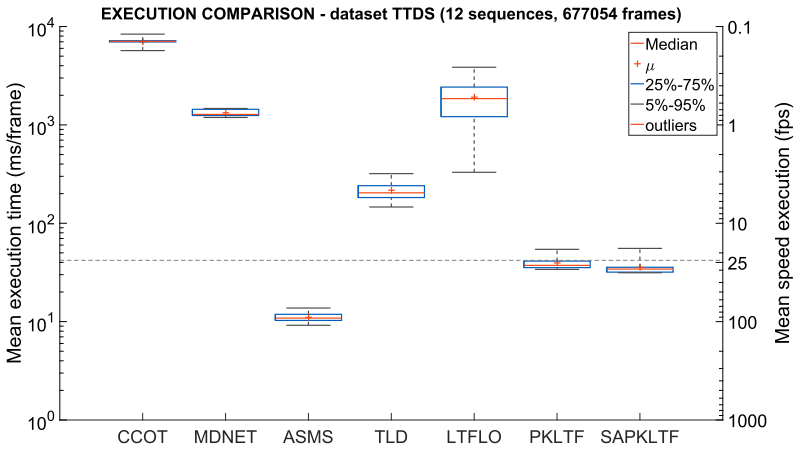
Figure B.20: TTds Center Error per sequence



a)

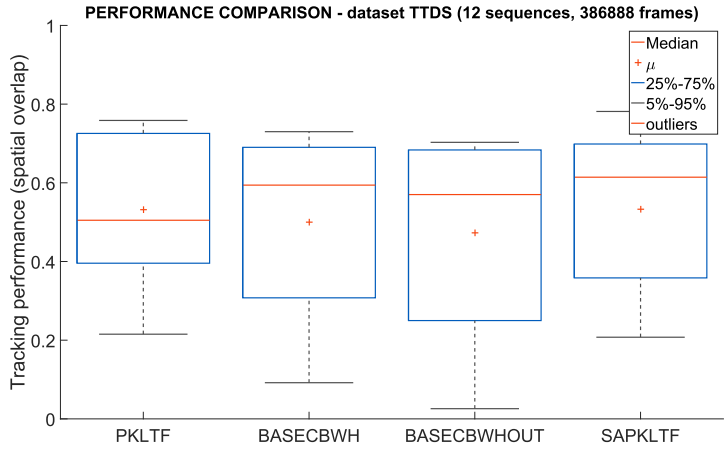


b)

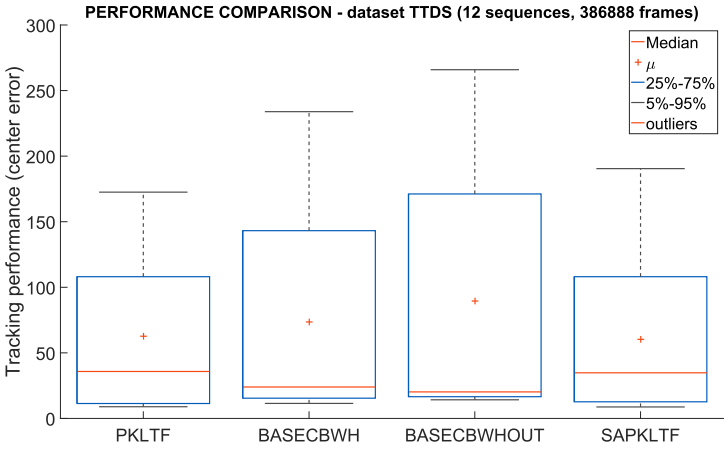


c)

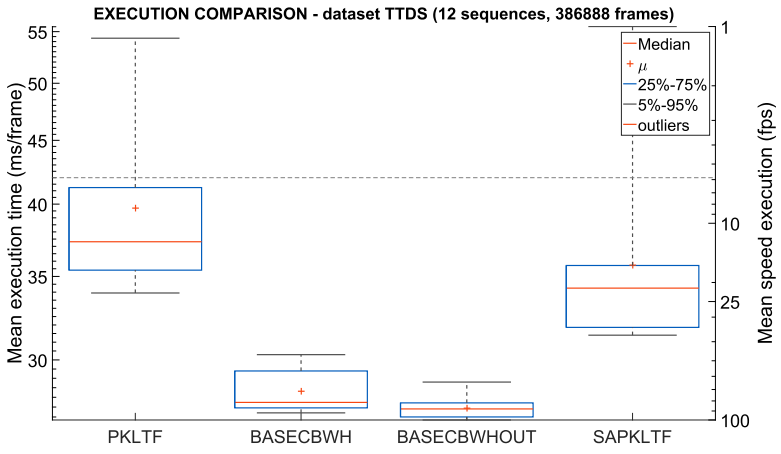
Figure B.21: TTds Comparison. a) Spatial Overlap; b) Center Error; and c) Execution Time



a)



b)



c)

Figure B.22: TTds Improvements Comparison. a) Spatial Overlap; b) Center Error; and c) Execution Time