Master Thesis

Visual Object Tracking applying Online Ensemble of multiple trackers

Author:
Jorge Martinez Gomez

Supervisor:
Juan Carlos Niebles

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science

in the

Computer Vision Research Group
Electrical and Electronics Engineering Department

May 2015

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

UNIVERSIDAD DEL NORTE

Abstract

ELECTRICAL AND ELETRONICS ENGINEERING DEPARTMENT
Computer Vision Research Group

Master of Science

Visual Object Tracking applying Online Ensemble of multiple trackers

by Jorge Martinez Gomez

The object tracking literature offers a large variety of tracking methods, which exhibit complementary properties in terms of their performance, best usage scenarios and failure modes. In this paper, we introduce a new tracking algorithm based on an online ensemble of tracking algorithms. Our method runs multiple online trackers in parallel and fuses their outputs in an online fashion. The resulting tracker can leverage the strenghts and overcome failures of each individual tracker, producing more robust target tracking. We perform experiments on current object tracking benchmark and show how our ensemble consistently outperforms all trackers in the ensemble, and achieves state-of-the-art object tracking performance.

Acknowledgements

In whatever you choose to do. Do it because it's hard, not because it's easy.

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

Dedicated to my future PS4

Chapter 1

Introduction

The goal of visual object tracking is to estimate the state of a target in an image sequence. This is a difficult task, as the target object can be articulated or deformable, the scene illumination can change suddenly, background clutter may introduce distractions that result in tracker drifting, among others. In spite of the multiple challenges, there are many potential applications that make this capability attractive such as activity recognition, motion analysis, human surveillance and robotics.

Many approaches for object tracking have been proposed to cope with some of these challenges. While the state-of-the-art methods achieve relative success, there is still no single approach that is able to handle all challenging situations. For instance, tracking-by-detection methods may not be able to handle scale variations rigorously. On the other hand, generative methods tend to suffer from model drifting and struggle to handle appearance variations.

In this paper, we focus on "model free tracking" of arbitrary objects in videos, in which no prior knowledge other than the object location in the first frame is available. Recently, the online tracking benchmark proposed in [**?**] shows that each tracking algorithm performs best under particular circumstances. There is no single tracking algorithm that can perform well on all sequences in the benchmark. This indicates that each tracking challenge can be addressed better by a different algorithm. In other words, tracking strenghts may be distributed among the available trackers. This is the key observation that inspires our proposed method; we consider a tracking approach that combines the outputs of multiple trackers running in parallel via an online ensemble. This ensemble has the interesting property of leveraging the strenghts of individual trackers, while overcoming the failure modes of each tracker. Since for a new and unseen sequence we do not know which tracker would perform best, our method computes a data-driven

online ensemble that results in improved tracking performance when compared to the results of individual trackers.

In our method, we leverage the observation that only some of the trackers drift into non-target areas of the image in most cases while some of the trackers succeed by focusing on the correct target. Furthermore, our ensemble uses an appearance model that serves as an additional verification mechanism of the tracked region. Using these model components, we identify and exploit the successful trackers to steer failed trackers towards the correct target region. Effectively, our ensemble can correct failed trackers, which ultimately increases tracking performance.

The main contribution of this paper is an ensemble tracking framework that builds on top of the output of available online tracking algorithms running in parallel to produce an online fused tracking result that leverages each tracker best features. Our method does not use prior knowledge about the nature of the trackers in the pool. The fused tracking output is obtained by considering appearance and spatial relations among tracker outputs. In order to cope with trackers weaknesses, our ensemble identifies successful trackers in a data-driven fashion and uses them to steer failed trackers by restarting them asynchronously. This helps to avoid sequence dependent parameters and overtuning.

The rest of the paper is organized as follows. We first briefly review the state-of-the-art of tracking algorithms in Section **??** and then present our online ensemble tracking algorithm in Section **??**. Section **??** illustrates quantitative and qualitative results of our tracker on a standard benchmarking dataset. Finally, we conclude the paper in Section **??**.

Chapter 2

Related Work

In this chapter, we provide a review of state-of-the-art tracking methods. Numerous approaches for object tracking have been proposed before. These methods differ from each other based on the way the authors solve common questions, such as: Which object representation is suitable for tracking the target? How can motion, appearance, or shape of the object needs to be modeled?. The answers for these questions depend on the context/environment where tracking is performed and the visual features of the object that needs to be tracked. This chapter is focused on presenting methodologies for object tracking in general and not for specific objects, such as humans or faces.

We follow a structure for describing the issues that are necessary to address when building an object tracker. In section 2.1 we describe common object representation and image features. Section 2.2 summarizes general schemes for detecting objects in an scenario. In Section 2.3, we categorize and describe existing tracking methods. Section 2.4 shows existing datasets used for object tracking and explains challenges which are present in different video sequences. Finally, we show in Section 2.5 recent tracking methods that perform ensemble of multiple trackers or features.

2.1 Object representation and visual features.

An object can be considered simply as nothing but an entity of interest used for further analysis. For instance, birds in the sky, pedestrians, vehicles on a road, ships on the sea, are set of objects necessary to track in a specific context. These elements can be represented by their shape or appearance. According to the object that is chosen to be tracked, an object representation is selected instead of the rest. For tracking small objects in a scenario, point representation is usually appropiate [1, 2]. In the case of

tracking objects whose shapes are approximated to rectangle and ellipses, shape representation is more appropiate [3]. To track objects with complex shapes, such as humans or excavators machines, contour or silhouette-based representation are appropiate [4].

In order to track objects, selecting the right features plays a critical role. In general, a feature is a property of an image which we are interested in. The most important property of a visual feature is its uniqueness so that could be easily distinguished from other objects. Mostly features are chosen manually by the user depending on the application domain. This problem of automatic feature selection has received significant attention in the pattern recognition community. The most common visual features selections are color [5, 6], edges [7, 8], displacement vectors [9, 10], corners [11] and textures [12–14].

Among all features, color is the one of the most widely used feature for tracking. However, color features are sensitive to illumination variation. To tackle this problem, in scenarios where this effect is inevitable, other features are incorporated to model the appearance of an object.

2.2 Moving Object Detection

Every tracking approach requires an object detection approach as initialization, or in every frame of the video. Detection can be defined as finding instances of objects in images or videos. A common method for object tracking is to apply object detection when the object appears for the first time, reducing the number of false detections.

A large number of methodologies have been proposed for object tracking, focusing on the task of object detection first. Most of them apply combinations among different methodologies, making it very difficult to create a uniform classification of existing approaches. This section classifies different approaches available for object detection from videos.

2.2.1 Background Subtraction

Background subtraction is a commonly used technique for object segmentation in static scenarios [15]. This task consists in detecting moving regions by subtracting the current image pixel-by-pixel from a reference background image. The pixels above some threshold are classified as foreground (belongs to an object). The background image is created averaging images over time in an initialization period, and is updated with new images to adapt to dynamic scene changes. Also, the foreground map is followed by morphological operations such as closing and erosion (elimination of small-sized blobs).

Although background subtraction techniques extracts well most of the relevant pixels, this method is sensitive to changes when some background and foreground pixels have similar value.

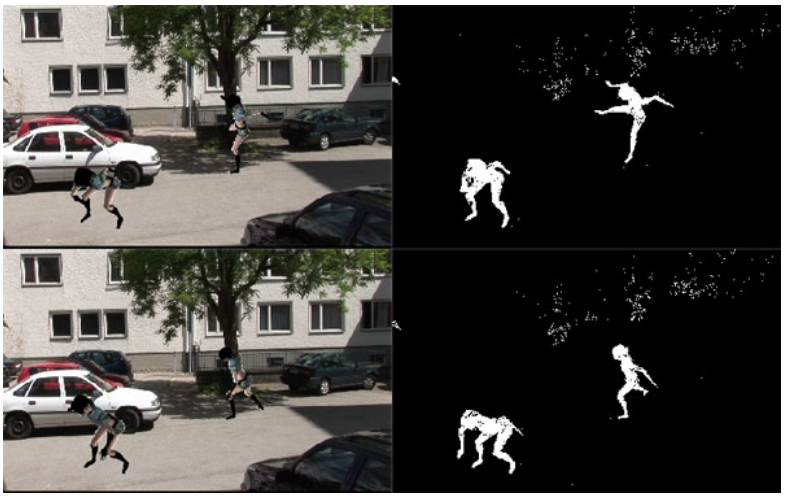


Figure 2.1: Object detection using Gaussian Mixture Models for background subtraction [16]. Foreground pixels are drawn in white.

2.2.2 Temporal differencing

In temporal differencing, objects are detected by taking pixel-by-pixel difference of consecutive frames (generally two or three) in a video sequence. This method is most common for object detection in scenarios where camera is moving. Unlike static camera scenarios, the background is changing rapidly (not appropiate to create a background model). Alternatively, the moving object is detected by taking the difference between frames $t-1$ and t.

This method is highly adaptive to dynamic changes in the scene as most recent frames are involved in the process. However, it fails detecting new small regions as moving objects (ghost regions). Detection will not be correct either, for objects that preserve uniform regions (static objects).

A two-frame differencing method is presented in [17], where the pixels that satisfy the following equation are marked as foreground.

$$|I_t(x,y), I_{t-1}(x,y)| > Th$$

Other methods were developed in order to overcome drastic changes of two frame differencing. For instance, a three-frame differencing method [18] and a hybrid method which combines three-frame differencing with an adaptive background subtraction model [19].

2.2.3 Statistical Approaches

Statistical characteristics of pixels have been used, in order to overcome shortcomings between frames of basic background subtraction methods. The approaches consist in keeping and updating pixels statistics that belong to the background model. Foreground pixels are identified by comparing each pixel's statistics with that of the background model. These methods are becoming more popular due to its reliability in scenes that contain noise, illumination changes and shadows. Some approaches apply Hidden Markov Models (HMM). These methods [20, 21] represent the intensity variation of a pixel in an image sequence as discrete states

The statistical method proposed in [16] describes and adaptive background model for real-time tracking. Every pixel is modeled by a mixture of Gaussians which are udpated online using incoming image data. Then, the Gaussians distributions of the mixture model for each pixel is evaluated in order to detect whether a pixel belongs to foreground and background.

2.2.4 Point detectors

Point detectors are used to find interesting points in objects which have an expressive texture in their respective localities. An interest point should have invariance to changes in illumination and camera viewpoint. One important detector uses optical flow (KLT) approach [22]. This method make use of the flow vectors of moving objects over time to detect moving blobs in an image. In this approach the apparent velocity and direction of every pixel in the frame must be computed. Some other methods are SIFT [23] and Harris [11] corners detectors.

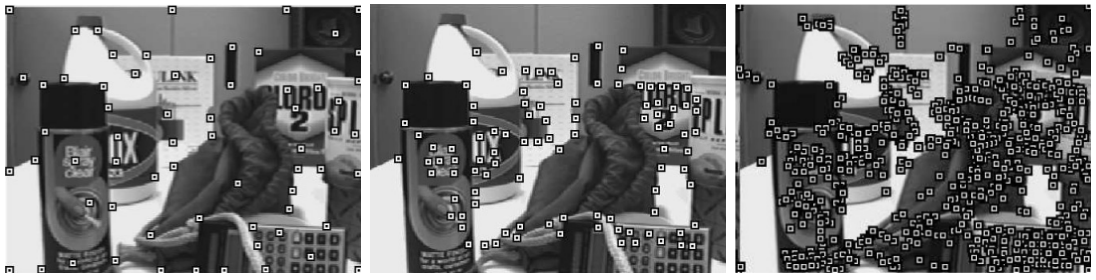


Figure 2.2: Interest points for object detection. Left - Harris, center - KLT, right - SIFT. Image taken from [24].

2.3 Object Tracking

The goal of an object tracker is to generate an object trajectory in a video. This path consists of the object position across the time. Additionally, a tracker may provides other informations, such as scale, orientation, area, or shape of an object. As stated in section 2.2, many combinations of object tracking approaches have been proposed before. This section classifies ans covers popular approaches on each category.

2.3.1 Point Tracking

Tracking can be formulated as the correspondence of objects represented by points across frames. This category can be divided into two subcategories:

Deterministic Methods. These approaches for point correspondence define a cost of associating each object in frame $t-1$ to a single object in frame t using motion constraints, such as proximity, velocity, rigidity and motion. Minimization of the correspondence cost is formulated as a combinatorial optimization problem. A solution, which consists in one-to-one correspondence among all possible associations, can be obtained by optimal assignment methods. For instance Hungarian Algorithm [25] or greedy search methods.

Statistical methods for Point Tracking. Statistical correspondence methods solve tracking problems whose measurements obtained from video sensors contain noise, or object motion can undergo random perturbations. These approaches take measurements and model uncertainties into account during object state estimation. Applying state space approach to modeling the object properties such as position, velocity and acceleration. In single object state estimation, tYhe optimal state of an object is given by a Kalman Filter [26, 27], assuming measurement noise have a Gaussian distribution. In the general case, that is, object state is not assumed as Gaussian, estimation can be performed using particle filters [21, 28].

In the case of multiobject data association, it is necessary to solve first correspondence problem before these filters can be applied. However, in cases when two objects are close each other, the correspondence could be incorrect. Then, an incorrectly associated measurement can cause the filter to fail to converge. In order to tackle this problem, Joint Probability Data Association Filtering (JPDAF) [29] and Multiple Hypothesis Tracking (MHT) [30] are two used techniques for data association.

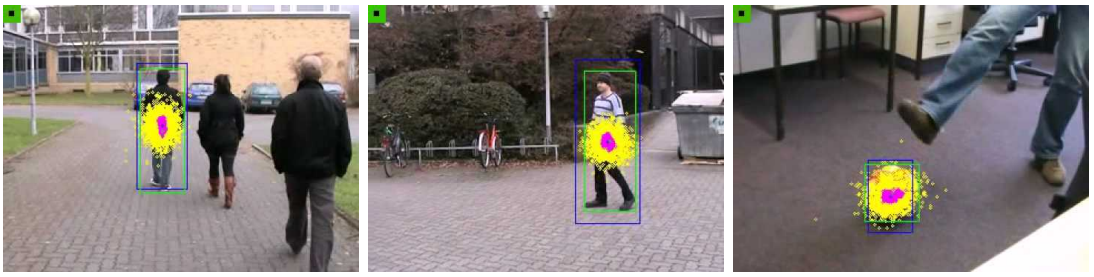


Figure 2.3: Object tracking using particle filter. Each particle represents one possible location of the object being tracked. A set of particles have more weight at locations where is more likely to be the object.Image taken from [21]

2.3.2 Kernel Tracking

In this type of tracking, the object motion is computed using representations of a primitive object region, from one frame to the next. These algorithms differ in terms of appearance representation, the number of objects to be tracked, and the method used for object motion estimation.

Density-based tracking: According to [31], the object is modeled with one or more probability density functions, such as Gaussian, mixture of Gaussian, Parzen windows or histograms, that describe the probability of object appearance. Mean-shift is an approach of density-based tracking. This method shifts a data point to the average of data points in its neighborhood, using fixed color distribution. A similar approach is called CAMSHIFT [32] that handles dynamically changing color distribution by adapting the search window size and computing color distribution in the search window.

Template-based tracking: These approaches apply templates of the object to calculate appearance probability on every frame of the video sequence. The most common is *Template matching* [33] that searchs accross the image, a region similar to the object template, defined in previous frames. The similarity measure is calculated using normalized cross correlation. A limitation of this method is its high computational cost due to brute force search. To reduce this cost, some methods limit the object search to a neighborhood near previous position.

Instead of templates, other object representations can be used for tracking. For example, color histograms or mixture models can be computed using the appearance of pixels inside a rectangular or ellipsoidal region. To reduce computational complexity, the similarity between object model and the hypothesized position, is computed evaluating the ratio between color means of object model and position. The position with highest ratio is selected as current object location.

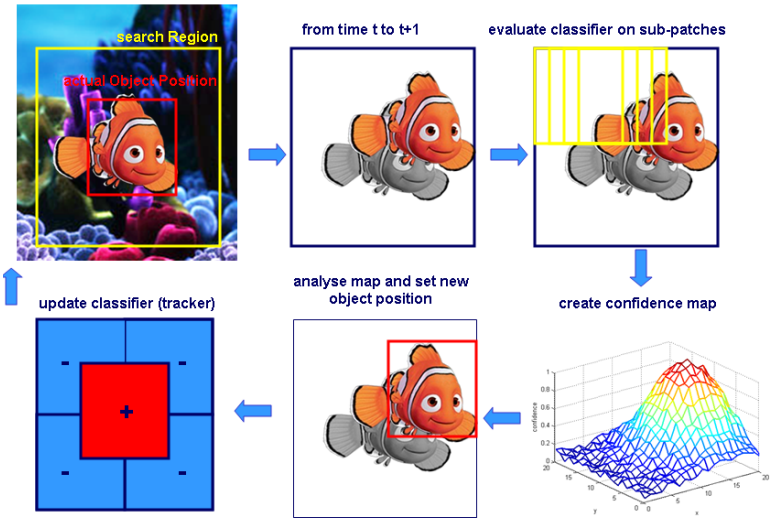


Figure 2.4: Overview for online boosting object tracker. For each classified patch, the system receives a confidence value, that is entered into a confidence map. Using the confidence map, the tracking windows is shifted to the best possible position. Then, the classifier is updated and the process is repeated.

"Tracking by detection" [34] systems generally perform target object appearance learning. These methods are closely related to object detection (an area with great progress in computer vision) and has encouraged some successful real-time tracking algorithms [35, 36]. However, many tracking algorithms employ static appearance models that are defined manually or trained at the first frame only [9, 37–40], these methods are often unable to deal with significant appearance changes. In order to cope this problem, an adaptive appearance model that changes during the tracking process as the appearance of the object changes, gets better results [41–43].

Boosting has been used in a wide field of machine learning tasks and applied to computer vision problems. Many tracking algorithms are based on the boosting framework [44] and is related to the work on Online Adaboost [45–47], multi-class boost [48] and MILBoost [49]. The goal of boosting is to combine many weak classifiers (usually decision stumps) into a linear strong classifier.

2.3.3 Silhouette Tracking

The object is tracked via estimation of the object region in each frame. Silhouette-based methods provide an accurate shape description for the objects that are tracked. These approaches can be divided into two main categories, shape matching and contour tracking. Shape matching [50] approaches search object silhouette in the current frame. Contour based, evolve initial contour to its new position using state space models or direct minimization of an energy function [51].

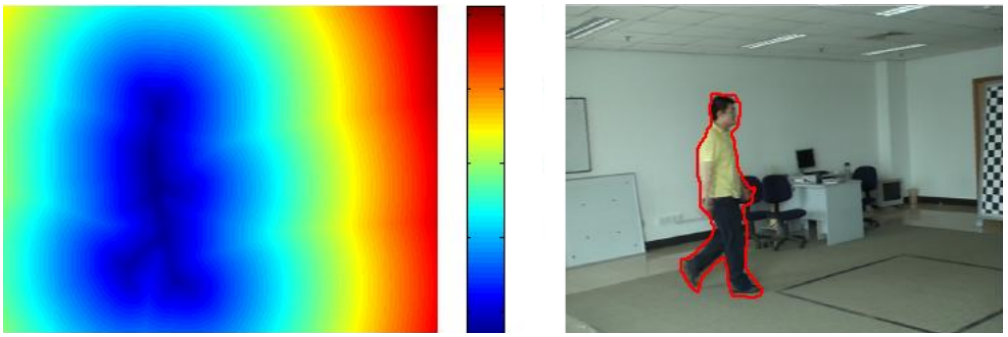


Figure 2.5: Illustration of an active contour representation. Left subfigure shows the signed distance map of a human contour; right image displays contour result. Image taken from [24]

2.4 Tracking Datasets and Challenges

Object detection and tracking is still an open research problem in computer vision. A robust, accurate and high performance approach is still a great challenge. The level of difficulty depends on how the object of interest is defined in terms of features. For instance, Using color as object representation method, it is not difficult to identify all pixels with same color as the object. However, there is always a probability of existence a background region with same color information (background clutter). In addition, illumination changes in the scene does not guarantee that the pixel values of an object will be the same in all frames. These variabilities or challenges which are random in object tracking causes wrong object tracking, and are listed below.

- **Illumination Variation (IV):** It is desirable that background model adapts to gradual changes of the appearance of the environment.
- **Scale Variation (SV):** Ratio between initial object size and current object size differs.
- **Occlusion (OC):** Partially or full, occlusion affects the process of computing the background frame. In real life situations, occlusion can occur anytime the object of interest passes behind another object with respect to a camera.
- **Dynamic background:** Some scenery regions contain movement, but should be still remain as background, acoording to their relevance. Such movement can be periodical or irregular, causing blurring (motion blur - MB), e.g. traffic lights, waving trees).
- **Out of view (OV):** Some portion o the target leaves the view.

- **Background clutter (BC):** As stated before, this challenge makes the segmentation task difficult. It is hard to create ans separate background model from moving foreground objects.
- **Fast Motion (FM):** The speed of a moving object plays an important role in its detection and track. If an object is moving too slow, the temporal differencing methods fails to detect object, because it preserves uniform region between frames. In the other case, fast moving object leaves ghost regions in a detected foreground model.
- **Object rotation and deformation (DEF):** Since natural objects move freely, they can appear slightly or completely transformed. Such rotations, in (IPR) or out (OPR) of plane on the images affect object tracking considerably.
- **Low Resolution (LR):** Number of pixels inside the object bounding box is less than 400.

Tracking Dataset: In computer vision, a *dataset* could be defined as a collection of images or video sequences used for testing algorithms. The amount of data and characteristics presented, depend on the field that is studied. For instance, in scene recognition, a dataset contains images of landscapes or outdoor environments. Generally, this collection is shared between researchers and plays an important role in comparison and evaluation of state-of-the-art approaches. A list of datasets used in object tracking is summarized in table 2.1

The Surveillance Performance Evaluation Initiative (SPEVI) [52] can be used for evaluating algorithms for surveillance-related applications. The first dataset contains 5 sequences applied to single person/face detection and tracking. The second dataset applies for multiple person/face detection and tracking. The sequences contain four targets occluding each other repeatedly. ETISEO dataset [53] contains indoor and outdoor scenes, such as corridors, buildings entries, etc. This dataset can be used for surveillance applications.

PETS dataset [54] became a surveillance project whose challenging scenarios are focused only on high level applications of this field. Some issues, like illumination or scale changes are not considered in these videos. Most of the sequences are used for person/vehicle tracking in outdoor environments(subway stations, building entrances). CAVIAR [55] is a dataset used generally for situation recognition systems. However, sequences can be applied for tracking evaluation methods. Includes videos of people walking alone, meeting other people, entering and exiting shops.

Dataset	# Sequences	GT-Available	Object
Bobot [59]	12	Yes	Generic
Cehovin [60]	5	Yes	Generic
Kalal [61]	10	Yes	Generic
Kwon [62]	4	Yes	Generic
Kwon VTD [63]	11	Yes	Generic
PROST [64]	4	Yes	Generic
Ross [41]	4	Yes	Generic
Thang [65]	4	Yes	Generic
Wang (NoRef)	4	Yes	Generic
Tracking Benchmark [57]	50	Yes	Generic
ALOV 300+ [58]	315	Yes	Generic
Godec [66]	7	Yes	Human
Babenko [49]	3	Yes	Human
SPEVI [52]	3,5	Yes	Human
ETISEO [53]	86	Yes	Human
PETS [54]	1,5,7,10	Yes	Human
CAVIAR [55]	6,11,6	Yes	Human
Clemson [67]	16	Yes	Human
VISOR [56]	6	No	Human

Table 2.1: Object tracking datasets.

The VIdeo Surveillance Online Repository (VISOR) [56] database covers a wide range of scenarios and situations, including videos for human action recognition, outdoor videos for face detection, indoor videos for people tracking with occlusions, vehicles detection and surveillance. The VIdeo Surveillance Online Repository, includes several sequences for two separate tasks: First, an abandoned baggage scenario and second, a parked vehicle scenario.

Recently, the tracking community released evaluation suites containing a selection of videos and algorithms for testing trackers performance. These benchmarks test and compare many tracking approaches using fair evaluation criteria. Online Object Tracking Benchmark [57] contais 50 most commonly used sequences. Also, the authors classified tracking challenges (attributes) into subsets to report specific challenging conditions. The Amsterdam Library of Ordinary Videos for tracking, ALOV300+ [58], consists in 315 real-life video sequences from Youtube with 64 different targets. The collection is categorized for thirteen aspects of difficulty and evaluates a large variety of situations including low contrast, occlusion, ilumination variation, etc.

2.5 Object tracking applying ensemble

Several methods have proposed the use of ensemble classifiers within the tracking-by-detection framework. In this perspective, instead of ensembling the outputs of other trackers, these methods focus on building ensemble classifiers, such as Adaboost [45] or Bayesian probabilistic fusion [68], from weak low-level image features. The classifier is then used to detect the target object in new frames. Instead of working directly with low-level image features, our ensemble tracker builds on top of tracker outputs which are used as a more powerful mid-level representation of the target.

Another alternative is to manually select a small set of trackers and use prior knowledge on the behaviour of each tracker to build an ensemble. In [64], the authors combine a template-based tracker, an optical flow tracker, and an online-random forest tracking-by-detection method into a cascade. In that case, the authors manually predefine a set of rules that decide how to ensemble the tracker outputs. In contrast, our method can handle a larger pool of trackers in the ensemble, since their outputs are combined in a data-drive fashion that does not require manual rules or prior knowledge on the behaviour of each tracker.

Sampling based approaches have also been explored for ensemble tracking. Examples of this are the VTD [69] and VTS [70] trackers. In this perspective, there is a sampling process that generates multiple samples of target and tracker states. These trackers run in parallel and their outputs are fused by probabilistic weighting. Therefore, the ensemble uses a set of trackers of similiar architecture with varying parameters. Our ensemble has the advantage of fusing outputs of multiple trackers with no assumptions on their architectural similarity and can leverage strenghts that different tracker architectures can provide.

Finally, one can consider the case of offline fusion of trackers, such as in [71], where all trackers are applied to the entire sequence and the ensemble is performed after the entire sequence is processed. However, we are interested in the case of online tracking, where the full sequence is not available beforehand. Furthermore, our online approach is capable of steering and reinitializing failed trackers, increasing the chances of better long-term tracking performance.

Chapter 3

Proposed approach

This section presents an overview of our online ensemble tracking approach, and then describe the details of each stage of our processing pipeline. Given tracking inputs, the algorithm estimates the trajectory of an object as it moves around a scene. The tracker outputs labels of the tracked object in every frame of the video 3.1. These labels provide position and scale information for evaluation and analysis.

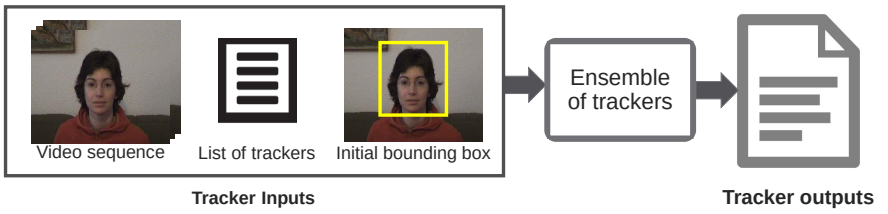


Figure 3.1: Tracking diagram. The proposed system takes initial bounding box, image sequences and the list of trackers as inputs. After tracking ensemble, the system drops out tracking results for each frame.

3.1 Overview

We illustrate the main components of our method in Figure 3.2. At each frame, our method proceeds as follows. First, we independently run all trackers $T = \{t_1, t_2, ..., t_n\}$ in a pool of size n, which produce a set of predicted target states $X = \{x_1, x_2, ..., x_n\}$ (Fig. 3.2a). These predictions are the raw input of our ensemble algorithm, and may be in the form of bounding boxes. A bounding box is here defines as the set of pixels covered by a rectangular area. Our ensemble methodology then looks for spatial coherence among the predicted target states, and verifies the appearance of the predicted image regions with respect to an object model (Fig. 3.2b). The idea is that for a given frame, we would

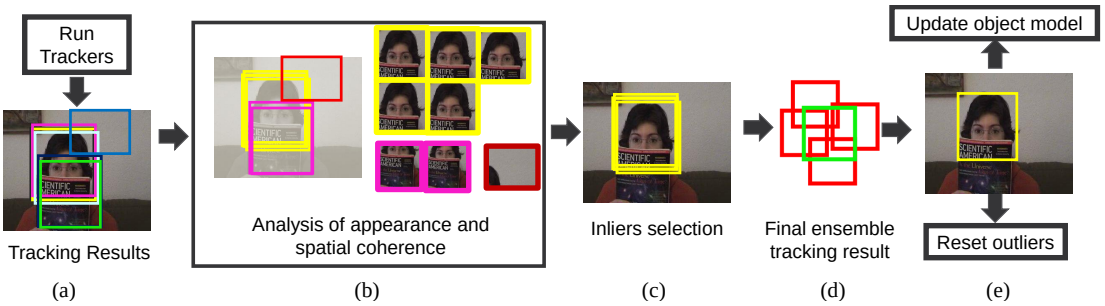


Figure 3.2: Basic diagram of our approach. On each frame, we analyze coherence between tracking results. We apply spatial and appearance models with the goal of finding inliers. Then, we select best tracker that follows the target. Finally, we reset outliers and update object model on necessary cases.

like to discover the subset of trackers that are correctly estimating the target state and to ignore all erroneous predictions. The result is a selection of inlier tracking predictions that contain the object with high confidence (Fig. 3.2c). A final ensemble prediciton of the target state is then derived from these inlier estimations (Fig. 3.2d). Finally, we use the prediction of the ensemble to update a model of the target appearance, and periodically reinitialize outlier trackers (Fig. 3.2e). This entire process is then repeated for each new frame in the video sequence.

In spite of the simplicity of our ensemble procedure, our experiments evidence that the ensemble is able to produce more accurate tracking results than any of the trackers in the pool. Furthermore, the ensemble can achieve state-of-the-art performance on benchmarking videos.

In the following, we provide the details of each processing stage and their role in the overall ensemble procedure.

3.2 Analysis of appearance and spatial coherence

After running all trackers in the pool, our method uses their state estimates X as input of the appearance and spatial coherence stage (Fig. 3.2b). The goal is to determine the set of trackers that correctly estimate the true target state. We denote these trackers as inliners in Fig. 3.2c.

In order to achieve this, we note a simple but key observation: in most cases, there is only a small subset of trackers that fail to correctly track the object in any given frame. In such scenario, the majority of the trackers focus correctly in the object, and we could use clustering techniques to automatically determine this set of inlier trackers. A more difficult scenario is when most of the trackers fail, but only a few focus in the correct image region. But even in this case, tracker failures tend to be distributed in varied

image locations, while the few correct trackers focus on a spatially coherent region. Once again, spatial clustering of the locations X can help discover the underlying true object location. In an extreme scenario, when only one tracker correctly follows the object, we can no longer rely on spatial clustering alone. A complementary cue can be introduced to overcome this issue: a target appearance model. We therefore explore the use of these two cues to select a subset of inlier trackers that follow the object region with high confidence.

3.2.1 Spatial clustering.

Cluster analysis is the formal study of algorithms and methods for grouping, or classifying objects. These objects are described as a set of measurements or by relationships between the object and other objects. A *cluster* is comprised of a number of similar objects collected or group together. Other authors define a cluster as a set of entities which are alike, and entities from different clusters are not alike, or "A cluster is an aggregation of points in the test space such that the *distance* between any two points in the cluster is less than the distance between any point in the cluster and any point not in it" [**?**].

Cluster analysis is the process of classifying objects into subsets that have meaning in the context of a particular problem. The objects are thereby organized into an efficient representation that characterizes the data. Clustering methods require that an index of proximity, or alikeness, or affinity, or association be established between pairs or patterns. A *proximity matrix* $|d(i,j)|$ accumulates the pairwise indices of proximity in a matrix in which each row and column represents a pattern. Diagonal entries of a proximity matrix are ignored since all paterns are assumed to have the same degree of proximity with themselves. Also it is assumed that all proximity matrices are symmetric, so all pairs of objects have the same proximity index, independent of the order in which they are written.A proximity index is either a *similarity* or a *dissimilarity*. The more

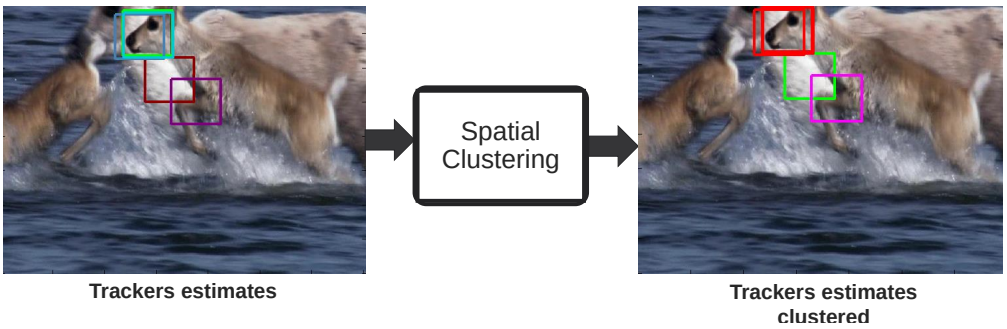


Figure 3.3: Spatial clustering stage. Using trackers estimate, we cluster similar bounding boxes using equation 4.1b. On right image, each color represents a cluster.

the ith and jth objects are similar one another, the larger a similarity index and the dissimilarity index are.

At this stage, we perform spatial clustering of all tracker predictions X. Bounding boxes with large overlap, similar location and scale should be grouped into the same cluster. Since we do not know the number of natural groups beforehand, we use an agglomerative clustering technique to achieve this. In practice, we apply complete-link hierarchical agglomerative clustering (CL) [**?**]. We define a dissimilarity measure between pairs of bounding boxes equal to:

$$d(x_i, x_j) = 1 - \frac{|x_i \bigcap x_j|}{|x_i \bigcup x_j|} \tag{3.1}$$

Using all dissimilarity values, we construct a symmetric $n \times n$ proximity matrix D. In CL, a pair of bounding boxes (x_i, x_j) will be grouped in the same cluster if their dissimilarity is below some threshold v. For all our experiments we set this value to 0.8. The result of CL is a grouping of the input bounding boxes X into m clusters $C = \{c_1, c_2, ..., c_m\}$, which are illustrated with colors in Fig. 3.2b. These clusters satisfy the following:

- $c_i \cap c_j = \emptyset$ for i and j from 1 to m, $i \neq j$
- $c_1 \cup c_2 \cup ... \cup c_m = T$

3.3 Object modeling.

Visual object tracking has been formulated as a tracking-by-detection problem recently. Object modeling is dynamically performed to support object detection over all frames. Mostly all the approaches can be classified into two categories: *Generative appearance models*, that mainly focus on how fit the data into their correspondent object class; and *discriminative appearance models*, which assume object tracking as a binary classification issue. Main goal is to maximize the separability between object and non-object regions discriminately.

Generally, Discriminative methods train a classifier using data acquired from previous frames, and subsequently use the trained classifier to evaluate possible object regions at the current frame. After localization, a set of *positive* and *negative* samples are heuristically selected to update the classifier. Some approaches apply online boosting [36, 46, 49], that make a discriminative evaluation of features taken from a candidate feature pool, and then select the top ranked features to conduct the tracking process.

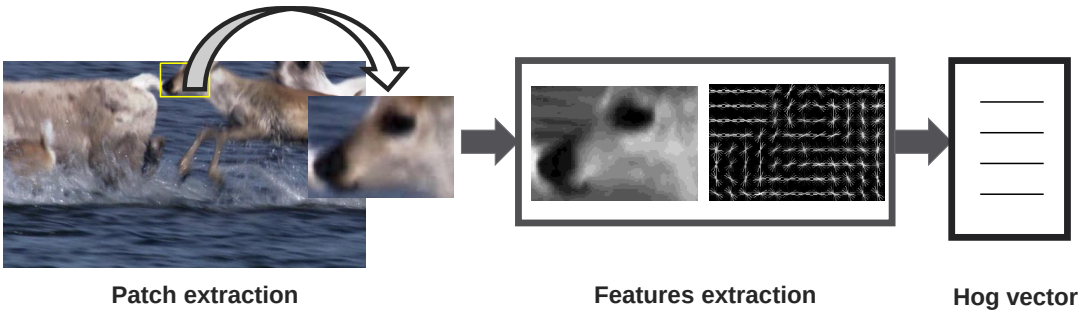


Figure 3.4: Patches and features extraction process. For a given patch, Histograms of Oriented Gradients are extracted. The final result, corresponds to a hog descriptor.

Other methods apply Support Vector Nachines (SVM) method, which learns a margin-based discriminative appearance model, in order to maximize inter-class separability. These classifiers are trained using visual representations of the object.

At this stage, we would like to verify the appearance of all tracker predictions X in comparison to an object appearance model. In order to do so, we train an appearance classifier that aims at separating positive target bounding boxes from background bounding boxes. We extract positive samples from the target location given at the first frame and also from uniformly sampling background bounding boxes around the target bounding box. In practice, we adopt a feature representation based on HOG and a SVM classifier with probability outputs. Using the classifier, we compute appearance scores $S = \{s_1, s_2, ..., s_n\}$ for each tracking result x_i in X (Figure 3.5). We considered this model, because other methods found in [68**?**], obtained better results. Also, HOG features are more robust to deformable objects than other image features used in tracking-by-detection methods *e.g.* Haar.

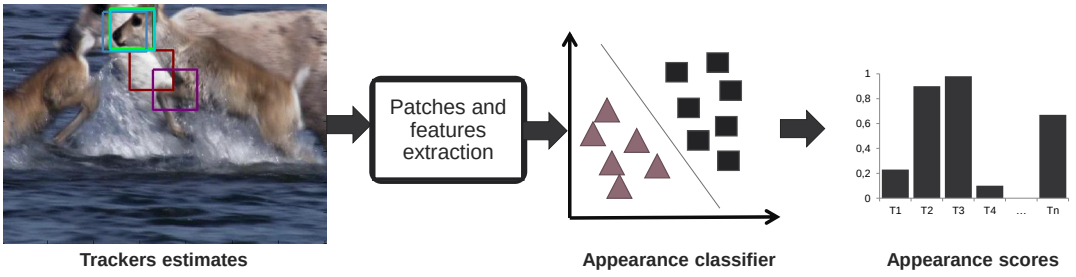


Figure 3.5: Appearance estimation for trackers estimates. On each frame, We compute appearance score for each tracker result using a previously trained classifier. We adopted HOG as image features.

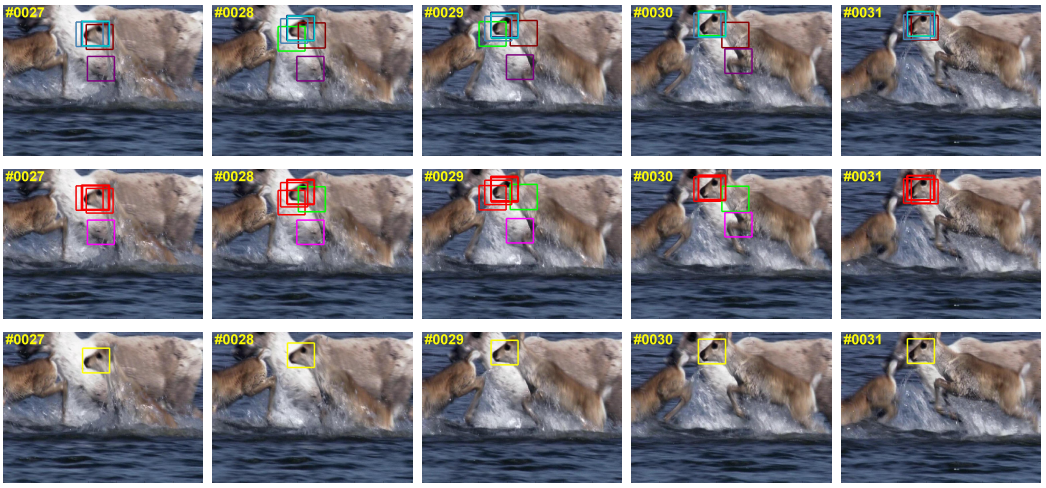


Figure 3.6: System behavior in five frames. Given image input, trackers will give results on where the object might be (first row). Then, all results are clustered using hierarchical agglomerative clustering (second row). We focus on selecting the group with highest number of members, or cluster with best appearance score (third row). Finally, we reinitialize outliers.

3.3.1 Selection of inliers

The clustering stage and appearance scoring provide cues about which trackers can be considered as correctly tracking the target. We evalute two simple criteria that optimally select a group of trackers as inliers (Fig. 3.2c). In our experiments, we consider cluster size and appearance scores as potential cues to select inliers.

Cluster size: This critera follows the idea that the largest spatially coherent cluster is associated to the true target location. Therefore, we flag all trackers in the largest cluster as inliners, and all other trackers as outliers.

$$c^* = \arg\max_{c_i \in C} |c_i| \tag{3.2}$$

Appearance scores: In this criteria, we trust our object appereance model to validate if a cluster contains bounding boxes that focus on the true target location. To do so, we max-pool the appereance scores s of the bounding boxes x within each cluster:

$$c^* = \arg\max_{c_i \in C} \max_{x_j \in c_i} s_j \tag{3.3}$$

3.3.2 Final ensemble tracking result

In order to provide an estimation of the state of the target using our ensemble, we fuse the outputs of all inlier trackers from the previous stage. There are multiple choices on how to implement this fusion. For example, it can be a linear combination of the inlier bounding boxes. In practice, we use a simpler approach that selects the medoid bounding box as the final ensemble output. That is, we output a bounding box x^* whose sum of distances with the rest of inlier bounding boxes is minimum:

$$x^* = \arg\min_{x_i \in c^*} \sum_{x_j \in c^*} d(x_i, x_j) \tag{3.4}$$

3.3.3 Model update and outlier reset

Once our ensemble estimates the target state, we can proceed to update the object model and steer failed trackers in the pool.

The first goal of this final stage is to keep our target appearance model updated. In order to avoid model drifting, we only update the appearance model periodically when our tracking ensemble has high confidence in the selection of inliers. Such confidence can be measured using the appearance scores S or the percentage of trackers in the inliner cluster. When confidence is high, a image patch is extracted at the predicted location and provided as a new positive sample to retrain or update our object appearance model.

The second goal is to steer failed trackers in the pool back to the target. When a tracker is consistently tagged as an outlier, our algorithm automatically resets its state to the latest target state prediction from the ensemble (Figure 3.7). This reinitialization procedure

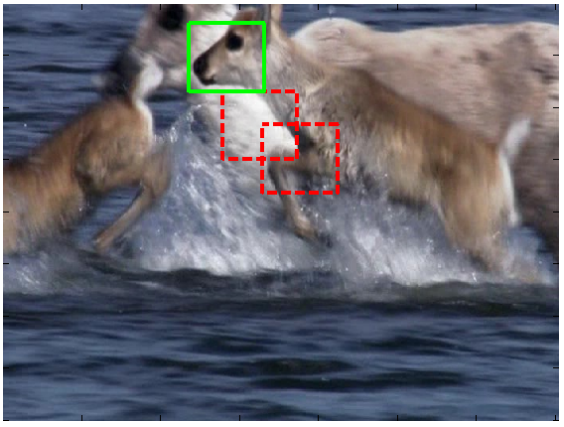


Figure 3.7: Outliers reinitialization. In this Figure, dashed red bounding boxes are tagged as outliers. These trackers states are reset to the latest target state prediction from the ensemble (green bounding box.)

is only performed periodically every 15 frames so that we can also take advantage of the capability of some trackers to recover from short-term tracking failures.

Chapter 4

Experiments

In this section we report evaluations for our proposed approach. We compare our tracking ensemble method and other state-of-the-art trackers using complete 50 sequences tracking benchmark [**?**]. We also present qualitative and quantitative analysis of our ensemble tracker, such as individual performance of trackers and effect of the tracker pool.

4.1 Implementation details

We implemented our online ensemble approach in MATLAB. CL clustering method is implemented in C++ and ported as *mex* function. Object model corresponds to a standard RBF SVM, implemented in LIBSVM library. To extract HOG features, we made use of Piotr Toolbox for Matlab using default parameters. All experiments were performed on a PC with an Intel Xeon CPU and 16 GB of RAM using same parameters over default dataset. This is commonly used to prevent over-tuning.

4.2 Evaluation Metrics

The trackers were evaluated using two evaluation metrics commonly used in literature. The first is *distance precision* (DP), which is the number of frames where the estimated center location is within a certain distance threshold d from the ground truth center location. The second metric is *overlap precision* (OP), defined as the number of frames where the óverlap between the estimated and ground truth bounding box exceeds a treshold b. For an image sequence, these two measures are calculated using equations 4.1a and 4.1b. Estimated center location and ground truth are denoted as $\mathbf{p}_f$ and

$\hat{\mathbf{p}}_f$ respectively, where f is the frame number. Futhermore, B_f and $\hat{B}_f$ denote the estimated and ground truth bounding boxes of the object. N is the number of frames in the sequence.

$$\mathrm{DP}(d) = \frac{1}{N}|\{f : ||\hat{\mathbf{p}}_f - \mathbf{p}_f|| \leq d\}|,\ d \geq 0 \tag{4.1a}$$

$$\mathrm{OP}(b) = \frac{1}{N}\left|\left\{f : \frac{|\hat{B}_f \bigcap B_f|}{|\hat{B}_f \bigcup B_f|} \geq b\right\}\right|,\ 0 \leq b \leq 1 \tag{4.1b}$$

In the recent literature, there has not been much agreement on which performance measure to use for comparing visual trackers. DP focuses only on estimated center location, which is beneficial for trackers that do not estimate scale of the object. Instead, OP also takes estimated scale into account and penalizes if the scale of the target is estimated incorrectly. We report results performance for both precision and overlap metrics.

Recently, authors of OOBT suggest the usage of precision and success plots. These curves show distance and overlap precision metrics over a range of thresholds. For instance, in precision plot, a higher precision at low thresholds means the tracker is more accurate, while a lost target will not achieve perfect precision on a large threshold range. In both types of plots, a *ranking score* is computed to line up trackers overall performance. In precision the DP value of 20 pixels is used as ranking score. In contrast to precision, in success plots, trackers are ranked using area under the curve (AUC), which relates to the average overlap across all frames.

To validate the performance of our proposed approach, we follow the one-pass evaluation methodology (OPE) proposed in [**?**]. We summarize the performance using the precision and success plots.

4.3 Dataset

In order to evaluate the performance of our proposed ensemble tracking, we adopt the Online Object Tracking Benchmark (OOTB) from [**?**]. This is an extensive benchmarking dataset that includes 50 sequences annotated with 11 attributes. The dataset includes challenging scenarios such as motion blur, illumination changes, scale variation, occlusions, in-plane and out-of-plane rotations, object deformation, background clutter and low resolution. The selection criterias of this dataset are listed below.

Table 4.1: Selected tracking algorithms for ensemble method. **Code Column**: M: Matlab, MC: Mixture of Matlab and C/C++, other: DLL files.

Method	**Type**	**Code**
Template matching - TM [**?**]	Template-based	other
Mean Shift - MS [**?**]	Point-based	other
Variance Ratio - VR [**?**]	Point-based	other
Peak Difference - PD [**?**]	Point-based	other
Ratio Shift - RS [**?**]	Point-based	other
Adaptive Structural Local Sparse Appearance		
Model - ASLA [**?**]	Sparse representation	MC
Compressive Tracker - CT [**?**]	Sparse representation	MC
Minimum Experts Entropy Minimization - MEEM [**?**]	tracking by detection	MC
Self-paced learning for long-term tracking - SPLTT [**?**]	tracking by detection	MC
Kernelized Correlation Filters - KCF [**?**]	Template-based	M
Dual Correlation Filters - DCF [**?**]	Template-based	M
Spatio-temporal Context Tracker - SCT [**?**]	Template-based	M
Circulant Structure Kernel - CSK, sKCF [**?**]	Template-based	M
Robust CSK tracker - RCSK [**?**]	Template-based	M
Minimum Output Sum of Squared Error - MOSSE [**?**]	Template-based	M

- For better comparison, OOTB videos are widely used as benchmark sequences in recent literature, allowing us to make a fair comparison to state-of-the-art algorithms.
- OOTB comes with ground-truth annotations for each sequence. This avoid labeling process.
- Each sequence corresponds to a set of image frames. This format allows to evaluate global processing time of the ensemble approach.
- Each benchmark sequence provide an annotation with 11 attributes, which explain the challenging scenarios encountered. This allow us to make attribute-based comparisons, which can show strengths and weaknesses of different trackers.

4.4 Tracker pool

Table 4.1 shows the list of the selected tracking algorithms. The selection criterias for the trackers are listed below.

- **Source code:** We integrate into our pool tracking algorithms whose original source code is publicly available. Executable files were not selected.
- **Store current state:** On each frame, a tracker must be able to store current state and give current result.

- **Trackers bounding box result:** All results must be given as a generic format (x,y, width, height), in order to perform ensemble.

4.5 Benchmarking results

We report the performance of our ensemble tracking algorithm on the 50 benchmarking sequences in Figure 4.1. The plots compare our approach with each individual tracker in Table 4.1 and state-of-the-art trackers: Struck tracker [**?**], Sparse Collaborative Model (SCM) [**?**], TLD tracker [61]. Our online ensemble tracking algorithm is denoted *OET*.

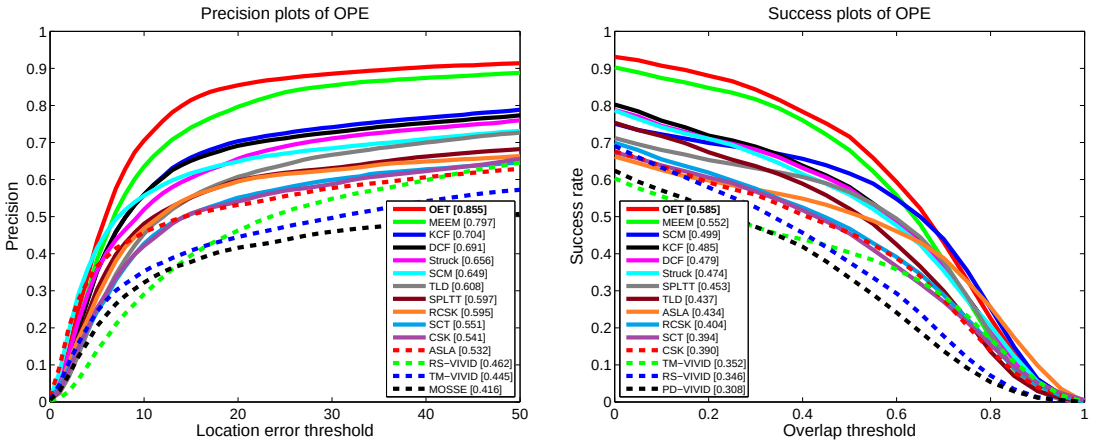


Figure 4.1: Precision and success plots for all 50 sequences. Precision and success ratios are measured by center location error and overlap ratio, respectively. Trackers are ranked using scores of 20 pixels for precision and AUC for success.

We note that our tracking ensemble improves performance over current state-of-the-art tracking algorithms in the benchmark. From Figure 4.1, our method outperforms each individual tracking score, in both precision and success. When inspecting individial sequences, we note that out method handles better camera rotation sequences, such as *singer1, singer2, fish, car4*, unlike the competing MEEM tracker. Also, our ensemble method can overcome complete object occlusion. In sequences where this situation happens *jogging, david3*, KCF trackers fails tracking the object.

4.6 Analysis of the tracking ensemble

In this subsection, we focus on the relevance of the number of trackers in the pool. We vary the size of the tracker pool in order to evaluate its effect on the performance of the ensemble. When selecting small tracker pools, we first chose the best 5 trackers from Fig. 4.1b. We then augment the pool with the next 5 trackers to form a pool of size 10.

Finally we add the worst 5 trackers for a pool of size 15. We also study the effect of the inlier selection criteria of Section 3.3.1 (appearance score and cluster size).

Table 4.2: Average AUC and precision for live fusion methods tested in 50 videos dataset.

Method	# trackers	Success(AUC)	Precision(20px)
Appearance score	5	0.585	0.855
	10	0.569	0.811
	15	0.560	0.804
Cluster size	5	0.528	0.765
	10	0.493	0.715
	15	0.480	0.678

The results are summarized in Table 4.2 for all 50 sequences in the benchmark. We report AUC ranking score for success, and 20-pixel threshold for precision. Our algorithm generally outperforms other methods using appearance score selection criteria. In case of cluster size selection, this method usually fails handling occlusions. In some sequences, many trackers lose object target and keep steady when occlusion happens, creating a big cluster which has the biggest number of members (Figure 4.2). However, this method is smoother and handles better fast motion and background clutter sequences.

We also note that adding trackers with lower performance hurts the ensemble. However, the drop in performance when adding weaker trackers, is less than 5% (∼300 frames) in success and 10% in precision (∼500 frames). For instance, when performing inlier selection using the appearance score criteria, a spurious tracker may focus on a region with very similar apperance to the object, *e.g.* background clutter, which can make tracking fail. This is very common in the *soccer* and *shaking* sequences.

In terms of running time, the average time cost for our ensemble method is 0.062 s/frame for 5 trackers, 0.122 s/frame for 10 trackers, and 0.198 s/frame for 15 trackers. These

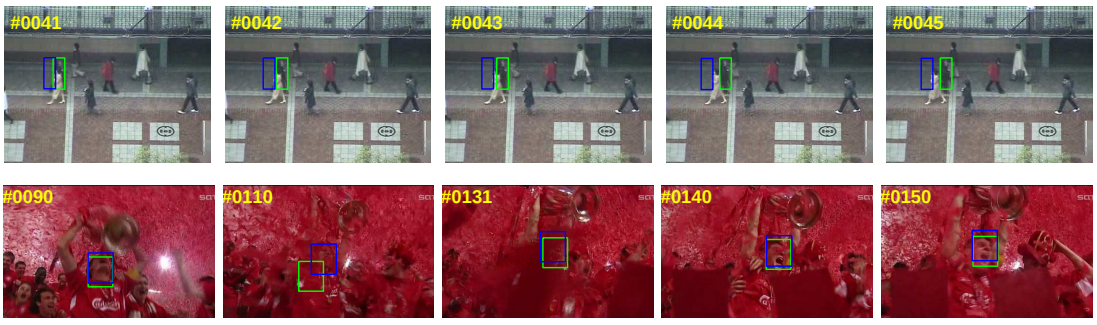


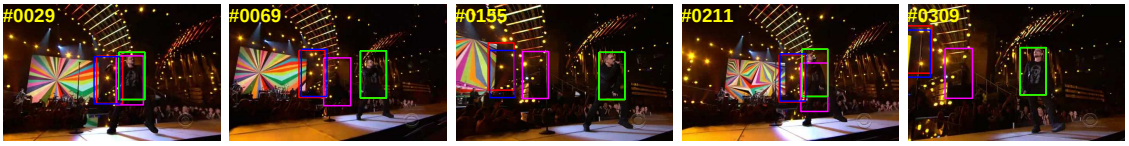
Figure 4.2: Qualitative results for object tracking applying both selection criterias in two sequences. **Green** bounding box corresponds to appearance score selection. **Blue** box to cluster size. In *subway* when occlusion happens, many trackers are lost, creating a big cluster. In cases of background clutter *soccer*, appearance selection loses target in some frames.

(a) bolt sequence.

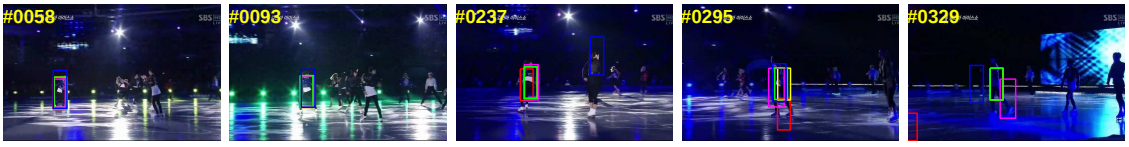
(b) jogging-2 sequence.

(c) singer-1 sequence.

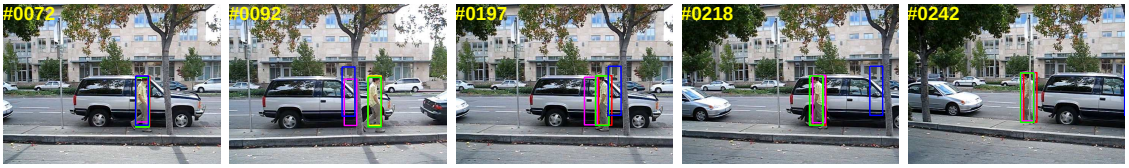
(d) football sequence.



(e) singer-2 sequence.



(f) skating1 sequence.



(g) david3 sequence.

Figure 4.3: Screenshots of tracking results.

timings do not include the processing time of each tracker.

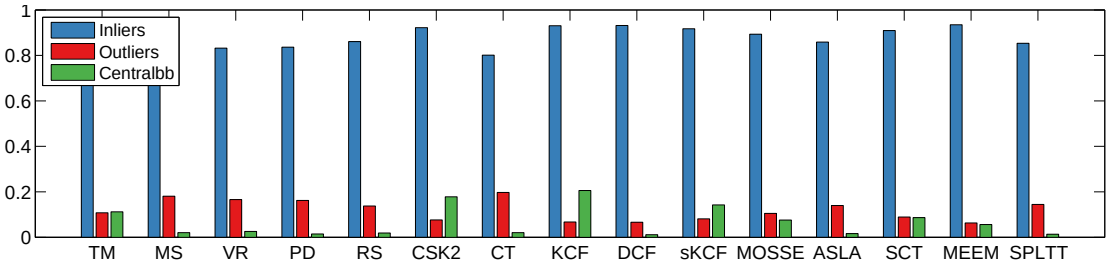


Figure 4.4: Statistics for each tracker in the ensemble over all frames.

On the other hand, we present statistics about how often trackers in the pool are selected as outlier, inliner, and medoid. In figure 4.4, for each tracker, there are 3 bars: percentage of frames that a tracker was in best cluster(inlier - blue bar); percentage of frames where a tracker was considered outlier (red bar); finally, percentage of frames that a tracker was selected as medoid bounding box (green bar). Based on this result, trackers whose individual performance is very high, have low percentage of being considered outliers (KCF, MEEM, RCSK). MEEM individual performance is very high. However, it does not have the highest frame percentage of being selected as central bounding box, in comparison with KCF or RCSK. Also, trackers that were considered spurious in previous experiments, have high rate in outliers bar (MS, VR, PD, RS).

4.7 Experiments with sequence attributes

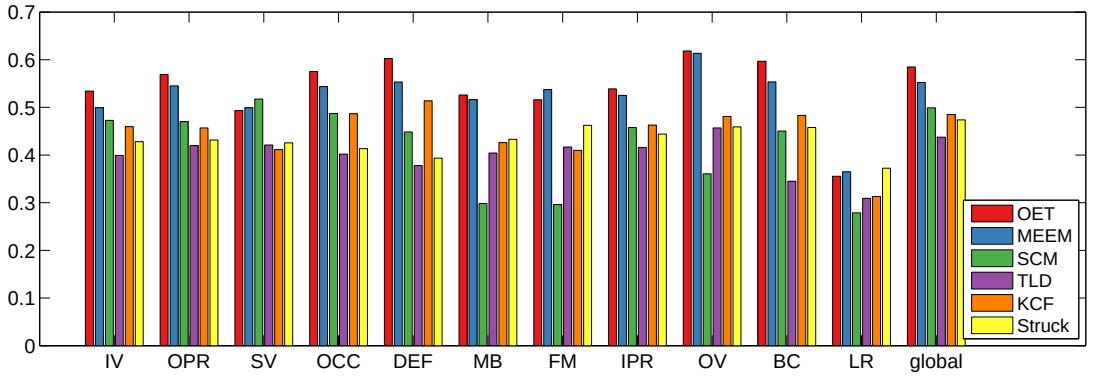


Figure 4.5: Average AUC ranking scores of top trackers on different subsets of test sequences in OPE. Each subset of sequences corresponds to an attribute: IV - illumination variation, OPR - out of plane rotation, SV- scale variation, OCC - occlusion, DEF - deformation, MB - motion blur, FM - fast motion, IPR - in plane rotation, OV - out of view, BC - background clutter, and LR - low resolution. Average AUC for all 50 videos is presented as global.

The videos in the benchmark dataset are organized and selected with attributes, which describe challenges present in the sequence - *e.g.* occlusion, object deformations. These

properties are useful for diagnosing tracking behavior, without the need of analyzing each video separately. Figure 4.5 shows AUC ranking scores of recent trackers on different sequences, grouped by attributes. For instance, background clutter (BC) contains all sequences whose target pixels might be confused with background.

From figure 4.5, our approach using appearance selection outperforms other trackers in 8 of 11 attributes. Specifically, in attributes such as IV (illumination variation), OPR (out of plane rotation), OCC (occlusion), DEF (deformation), MB (motion blur), IPR (in plane rotation), OV (out of view), and BC (background clutter). It is important to note that SCM tracker is better than most recent trackers in terms of scale variation. Results show that its affine motion models handle scale variation better than other trackers, which are designed to account translational motion [**?**]. In our system, scale is not determined. We are dependent of each separated tracker scale, and some trackers do not consider scale correction. Some of them apply initialization scale over all sequence.

Bibliography

[1] Cor J. Veenman, Marcel J T Reinders, and Eric Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:54–72, 2001. ISSN 01628828.

[2] Khurram Shafique and Mubarak Shah. A noniterative greedy algorithm for multiframe point correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:51–65, 2005. ISSN 01628828.

[3] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–577, 2003. ISSN 01628828.

[4] Ismail Haritaoglu, David Harwood, and Larry S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:809–830, 2000. ISSN 01628828.

[5] G. Paschos. Perceptually uniform color spaces for color texture analysis: An empirical evaluation. *IEEE Transactions on Image Processing*, 10:932–937, 2001. ISSN 10577149.

[6] K. Y. Song, J. Kittler, and M. Petrou. Defect detection in random colour textures. *Image and Vision Computing*, 14:667–683, 1996. ISSN 02628856.

[7] J Canny. A computational approach to edge detection. *IEEE transactions on pattern analysis and machine intelligence*, 8:679–698, 1986. ISSN 0162-8828.

[8] Kevin Bowyer, Christine Kranenburg, and Sean Dougherty. Edge Detector Evaluation Using Empirical ROC Curves. *Computer Vision and Image Understanding*, 84:77–103, 2001. ISSN 10773142.

[9] Michael J Black and Allan D Jepson. EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. *International Journal of Computer Vision*, 26:63–84, 1996. ISSN 0920-5691.

[10] Bruce D Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. *Imaging*, 130:674–679, 1981. ISSN 17486815.

[11] C. Harris and M. Stephens. A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, pages 147–151, 1988. ISSN 09639292.

[12] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3, 1973. ISSN 0018-9472.

[13] Kevin M. Nickels and Seth Hutchinson. Textured image segmentation, 1997. ISSN 02628856.

[14] Stephane G. Mallat. Theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693, 1989. ISSN 01628828.

[15] Am McIvor. Background subtraction techniques. *Proc. of Image and Vision Computing, ...*, 2:13, 2000.

[16] Vu Pham, Phong Vo, Vu Thanh Hung, and Le Hoai Bac. GPU Implementation of Extended Gaussian Mixture Model for Background Subtraction. *2010 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, pages 1–4, 2010.

[17] A J Lipton, H Fujiyoshi, and R S Patil. Moving target classification and tracking from real-time video. *Proceedings Fourth IEEE Workshop on Applications of Computer Vision WACV98 Cat No98EX201*, 98:8–14, 1998. ISSN 09031936.

[18] Liang Wang, Weiming Hu, and Tieniu Tan. Recent developments in human motion analysis. *Pattern Recognition*, 36:585–601, 2003. ISSN 00313203.

[19] Robert T Collins, Alan J Lipton, Takeo Kanade, Hironobu Fujiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, Osamu Hasegawa, Peter Burt, and Lambert Wixson. A System for Video Surveillance and Monitoring, 2000. ISSN 19406029.

[20] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J.M. Buhmann. Topology free hidden Markov models: application to background modeling. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 1, 2001.

[21] J Rittscher, J Kato, S Joga, and A Blake. A probabilistic background model for tracking. *Computer Vision—ECCV 2000*, pages 336–350, 2000.

[22] Jianbo Shi Jianbo Shi and C. Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, 1994. ISSN 1063-6919.

[23] David G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. pages 1–28, 2004.

[24] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38:13, 2006. ISSN 03600300.

[25] Zhen Qin and Christian R. Shelton. Improving multi-target tracking via social grouping. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1972–1978, 2012.

[26] Xiaofeng Ren. Finding people in archive films through tracking. In *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.

[27] Janne Heikkilä and Olli Silvén. A real-time system for monitoring of cyclists and pedestrians. *Image and Vision Computing*, 22:563–570, 2004. ISSN 02628856.

[28] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little, and David G Lowe. A Boosted Particle Filter : Multitarget Detection and Tracking. *Proceedings of the 8th European Conference on Computer Vision - ECCV 2004*, pages 28–39, 2004. ISSN 03029743.

[29] Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. People Tracking with Mobile Robots Using Sample-Based Joint Probabilistic Data Association Filters, 2003. ISSN 02783649.

[30] Mohd Asyraf Zulkifley, Bill Moran, and David Rawlinson. Robust hierarchical multiple hypothesis tracker for multiple object tracking. In *Proceedings - International Conference on Image Processing, ICIP*, pages 405–408, 2012.

[31] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:790–799, 1995. ISSN 01628828.

[32] David Exner, Erich Bruns, Daniel Kurz, Anselm Grundhöfer, and Oliver Bimber. Fast and robust CAMShift tracking. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*, pages 9–16, 2010.

[33] Simon Korman, Daniel Reichman, Gilad Tsur, and Shai Avidan. FasT-match: Fast affine template matching. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2331–2338, 2013.

[34] Greg Mori and Jitendra Malik. Recovering 3D human body configurations using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 28:1052–1062, 2006. ISSN 01628828.

[35] Xiaoming Liu and Ting Yu. Gradient feature selection for online boosting. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[36] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-Time Tracking via On-line Boosting. *Technology*, 1:1–10, 2006. ISSN 0162-8828.

[37] M. Isard and J. MacCormick. BraMBLe: a Bayesian multiple-blob tracker. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 2, 2001.

[38] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1465–1479, 2006. ISSN 01628828.

[39] D Comaniciu, V Ramesh, and P Meer. Real-time tracking of non-rigid objects using mean shift. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:142–149, 2000. ISSN 01628828.

[40] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 798–805, 2006.

[41] David a. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental Learning for Robust Visual Tracking. *International Journal of Computer Vision*, 77:125–141, 2007. ISSN 0920-5691.

[42] Iain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:810–815, 2004. ISSN 01628828.

[43] A D Jepson, D J Fleet, and T F El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1296–1311, 2003. ISSN 0162-8828.

[44] Y Freund and R E Schapire. A Decision-theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of Computing Systems and Science*, 55:119–139, 1997. ISSN 00220000.

[45] Shai Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:261–271, 2007. ISSN 01628828.

[46] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5302 LNCS, pages 234–247, 2008.

[47] NC Oza and S Russell. Online ensemble learning. *AAAI/IAAI*, 6837:1109–1109, 2000.

[48] Amir Saffari, Martin Godec, Thomas Pock, Christian Leistner, and Horst Bischof. Online multi-class LPBoost. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3570–3577, 2010.

[49] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual Tracking with Online Multiple Instance Learning. *IEEE transactions on pattern analysis and machine intelligence*, pages 983–990, 2010. ISSN 1939-3539.

[50] Baoxin Li, Rama Chellappa, Qinfen Zheng, and Sandor Z. Der. Model-based temporal object verification using video. *IEEE Transactions on Image Processing*, 10: 897–908, 2001. ISSN 10577149.

[51] Daniel Cremers and Christoph Schnörr. Statistical shape knowledge in variational motion segmentation. *Image and Vision Computing*, 21:77–86, 2003. ISSN 02628856.

[52] Emilio Maggio and Andrea Cavallaro. Hybrid particle filter and mean shift tracker with adaptive transition model. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, volume II, 2005.

[53] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1863–1868, 2006. ISSN 01628828.

[54] Alexandre Alahi, Laurent Jacques, Yannick Boursier, and Pierre Vandergheynst. Sparsity-driven People Localization Algorithm: Evaluation in Crowded Scenes Environments. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, Snowbird, Utah, 2009.

[55] A. Torralba, K.P. Murphy, W.T. Freeman, and M.A. Rubin. Context-based vision system for place and object recognition. *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003. ISSN 0769519504.

[56] Roberto Vezzani and Rita Cucchiara. Video surveillance online repository (ViSOR): An integrated framework. *Multimedia Tools and Applications*, 50:359–380, 2010. ISSN 13807501.

[57] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online Object Tracking: A Benchmark. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, (Iccv): 2411–2418, June 2013.

[58] Arnold W M Smeulders, Dung M. Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:1442–1468, 2014. ISSN 01628828.

[59] Dominik A. Klein, Dirk Schulz, Simone Frintrop, and Armin B. Cremers. Adaptive real-time video-tracking for arbitrary objects. In *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 772–777, Oct 2010.

[60] Luka Cehovin, Matej Kristan, and Ales Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE transactions on pattern analysis and machine intelligence*, 35:941–53, 2013. ISSN 1939-3539.

[61] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-Learning-Detection. *IEEE transactions on pattern analysis and machine intelligence*, 34:1409–1422, 2011. ISSN 1939-3539.

[62] Junseok Kwon and Kyoung Mu Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *CVPR*, pages 1208–1215, 2009.

[63] Junseok Kwon and Kyoung Mu Lee. Visual tracking decomposition. In *CVPR*, pages 1269–1276, 2010.

[64] Jakob Santner, Christian Leistner, Amir Saffari, Thomas Pock, and Horst Bischof. PROST: Parallel robust online simple tracking. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 723–730, 2010.

[65] Thang Ba Dinh, Nam Vo, and Gérard Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1177–1184, 2011.

[66] Martin Godec, Peter M. Roth, and Horst Bischof. Hough-based tracking of non-rigid objects. In *Proc. International Conference on Computer Vision (ICCV)*, 2011.

[67] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 232–237, 1998.

[68] Qinxun Bai, Zheng Wu, S Sclaroff, M Betke, and C Monnier. Randomized Ensemble Tracking. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2040–2047, 2013.

[69] Junseok Kwon and Kyoung M. Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pages 1208–1215, 2009.

[70] Junseok Kwon and Kyoung Mu Lee. Tracking by sampling trackers. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1195–1202, 2011.

[71] Christian Bailer, Alain Pagani, and Didier Stricker. A Superior Tracking Approach: Building a strong Tracker through Fusion. In *European Conference on Computer Vision*, pages 170–185, 2014.