Robust visual tracking using feature selection

Master thesis

Fabien Willemin
epfl ee lts2 – 2013
fabien.willemin@epfl.ch

Supervised by:

Pierre Vandergheynst – EPFL
Signal Processing Lab
http://lts2www.epfl.ch

Johan Paratte – EPFL
Signal Processing Lab
http://lts2www.epfl.ch

Abstract

Visual tracking has become a very important component in computer vision, but achieving a robust, reliable and real time tracking remains a real challenge.

In order to improve the actual state-of-the-art, we choose to study and improve one of the most performing adaptive tracker by detection. We selected Struck [27] for this quality performance and his low computational cost that makes it real time.

Inspired by the great successes of binary keypoint descriptors, we choose to apply binary description to a patch. We propose to use Multi-Block Local Binary Pattern (MB-LBP), based on its great success in face detection and description. In this work we present a technique for selecting the best features for tracking. In combination with the feature selection we propose a technique to take into account contextual information in order to increase the robustness of the tracker.

We propose a solution to add scale adaptation to the algorithm, and suggest to transpose this technique to add rotation adaptation.

Experimentally we validate these techniques showing that we outperform the state-of-art tracking algorithms. To do that we use a benchmarking tool using 51 videos and compare our algorithm to 29 algorithms.

Contents

Acronyms

BRISK Binary Robust Invariant Scalable Keypoints

BRIEF Binary Robust Independant Elementary Features

ORB Oriented FAST and Rotated BRIEF

FREAK Fast Retina Keypoint

HOG Histograms of Oriented Gradients

LBP Local Binary Patterns

MB-LBP Multi-Block LBP

SIFT Scale-Invariant Feature Transform

SURF Speeded-Up Robust Features

SVM Support Vector Machine

Struck Structured Output Tracking with Kernels

Introduction

Due to the low cost and the increasing quality of the visual sensors, they have become a good choice for the motion estimation of objects. The noninvasive aspect makes video cameras very versatile, but the drawback is the poor reliability. Having a robust and reliable visual tracking using cameras can have great applications and at the same time hard to achieve, and this makes the subject very interesting.

This task is indeed a real challenge, and the reason is that even if tracking a semantic object is relatively obvious for a human, the problem is not easy to define in term of images. The recent evolution in computing power and computer vision has made possible to tackle this challenge only in recent years. The importance of having a really reliable, robust and precise tracking has stimulated the current research in the area [83]. The scope of the domain of applications is wide and we can cite surveillance [28, 13, 31], human computer interaction [63, 75, 33], medical imaging [22] and robotics [32, 78].

In this document we only focus on online (i.e. realtime) tracking. That is, given the initialized state (e.g. rectangle position and size) of an object in a frame of a video, the goal of online tracking is to estimate the position of the target in the next frames. There is no need of having the entire video stream, but only the frames between the initial frame with the positioned target and the frame where we want to know the position of the target.

The algorithm need to deal with illumination variation, partial or total occlusions and change in object appearance. An algorithm can be really robust to one kind of change but not to another, so it is crucial to have both a good dataset and a good evaluation method to take this variability in consideration.

Part I
Theoretical Foundations

1 Visual tracking

1.1 Purpose

The challenge is to design an algorithm that is reliable and robust to a wide variety of conditions, particularly if the camera is mobile. The algorithm need to deal with illumination variations, occlusions, and appearance changes. The motions of the object and camera are obviously unknown so it is impossible to deal with all the parameters without assumptions. The goal is thus not to make too strong assumptions.

1.2 Techniques

In this section we present some tracking techniques, focusing on the current state-of-the-art, i.e tracking by detection with an adaptive appearance model. There are however many other techniques and we quickly present a subset in the following.

1.2.1 Snakes models [38]

A snake is an energy-minimizing spline guided by external constraint forces and influenced by image forces that pull it toward features such as lines and edges. Snakes are active contour models: they lock onto nearby edges, localizing them accurately. They are really powerful for lane tracking [72] and for autonomous car or driver assistance. They are however not not very efficient for tracking arbitrary objects (without prior model) and deal poorly with occlusions.

1.2.2 Condensation [34]

The Condensation algorithm uses "factored sampling", previously applied to the interpretation of static images, in which the probability distribution of possible interpretations is represented by a randomly generated set. Condensation uses learned dynamical models, together with visual observations, to propagate the random set over time.

1.2.3 Mean shift [15, 16]

Mean shift is based on the mean shift iterations and finds the most probable target position in the current frame. The dissimilarity between the target model (its color distribution) and the target candidates is expressed by a metric derived from the Bhattacharyya coefficient.

1.3 Tracking by detection [6]

Recently, the techniques called “tracking by detection” have been shown to give promising results at realtime speeds. These methods train a discriminative classifier in an online manner to separate the object from the background. The classifier bootstraps itself by using the current tracker state to extract positive and negative examples from the current frame. This technique is organized around three main components : representation scheme, search mechanism and model update.

1.3.1 Representation Scheme

Object representation is one of the major components in any visual tracker and is really crucial for the efficiency of the algorithm. Various techniques have been proposed [44, 74],

and one of the first method is simply based on holistic templates (i.e. raw intensity values) [2, 26, 50]. This simple representation does not take appearance changes into account, so more sophisticated models have been proposed, such as subspace-based tracking approaches [10, 4]. In addition sparse representations have been proposed to handle corrupted appearances in [51] whose method has recently been further improved in [8, 52, 76, 82, 71, 53].

In addition to templates, many other visual features have been adopted in tracking algorithms, such as color histograms [16], Histograms of Oriented Gradients (HOG) [18, 65], covariance region descriptors [67, 73, 57] and Haar-like features [70, 23].

Discriminative models give very good results and they are widely adopted in tracking [4, 14]. In short, a binary classifier is learned online to discriminate the target from the background. The majority of learning techniques are not usually online, but for obvious performance matter, it is really important to find online implementations, that give the same results or a good approximation of the original offline implementations. Numerous learning methods have been adapted to the tracking problem, such as Support Vector Machines (SVM) [3, 29], structured output SVM [27, 66], ranking SVM [7], boosting [4, 23], semi-boosting [24] and multi-instance boosting [5].

To better cope with appearance variations, some approaches regarding integration of multiple representation schemes have recently been proposed [45, 64, 39].

1.3.2 Search Mechanism

To estimate the position of the target objects, deterministic or stochastic methods have been used. When the tracking problem is posed within an optimization framework, assuming the objective function is differentiable with respect to the motion parameters, gradient descent methods can be used to locate the target efficiently [48, 16, 20, 41].

However, the problem is usually non-linear and contain many local minima. To alleviate this problem, dense sampling methods have been adopted [27, 23, 5] at the expense of high computational load. In practice, it really depends on the cost of a sample evaluation, but it can quickly become computationally expensive.

On the other hand, stochastic search algorithms such as particle filters [34, 56] have been widely used since they are relatively insensitive to local minima and computationally efficient [36, 58, 51]. The advantage of these methods is to reduce the cost of a dense sampling method (when it is necessary) without missing the optimal solution.

1.3.3 Model Update

It is crucial to update the target representation or model to account for appearance variations. It is a really difficult task, because we need to adapt the model without introducing drifts, or falsely adapt the model during an occlusion. Effective update algorithms have also been proposed via online mixture model [35], online boosting [23] or incremental subspace update [58]. For discriminative models, the main issue has been to improve the sample collection part to make the online-trained classifier more robust [27, 5, 37, 24]. While a lot of progress has been made, it is still difficult to get an adaptive appearance model that avoids drifts, especially if the video is long. Indeed most of sequences used for benchmarking are pretty short (a few seconds), so drifts are not necessary noticeable for all algorithms, although it remains a really challenging task.

1.3.4 Context and Fusion of Trackers

Contextual information is also very important for tracking. It is important to avoid drifts and handle full occlusions correctly. Recently some approaches have been proposed by mining auxiliary objects or local visual information surrounding the target to assist the tracking [77, 25, 19]. The context information is especially helpful when the target is fully occluded or leaves the image region [25]. To improve the tracking performance, fusion methods have been proposed. This approach combines static, moderately adaptive and highly adaptive trackers to account for appearance changes [62]. Even multiple trackers [40] or multiple feature sets [79] are maintained and selected in a Bayesian framework to better account for appearance changes. We will see in this work how the extension of the target size (i.e. accounting for contextual information) around the object can improve the tracking.

2 Patch description

2.1 Purpose

The first technique and also the simplest is a holistic template (i.e. raw intensity values) [2, 26, 50]. It has however some weaknesses, such as:

- Very high dimentionality
- Very sensitive to illumination variations
- Very sensitive to all geometric transformations
- Very sensitive to occlusions

An alternative is the patch description whose purpose is to find a way to describe a patch by avoiding most of the above drawbacks, but with a relatively low computational cost.

There are two families of patch descriptors : feature-point descriptors (or keypoints descriptors) and patch descriptors. The former describes a point and its neighborhood and the latter describes a patch as a whole rectangle.

2.2 Feature-point description

In the keypoint description framework, two families of descriptors can be distinguished : the floating point descriptors and the binary descriptors.
The floating point descriptor is older and a little bit more descriptive but come with a very high computational cost. Scale Invariant Features Transform (SIFT) [47] and then Speeded-Up Robust Features (SURF) [9] are the most famous ones. Their computational cost make them not very attractive for tracking.

Binary descriptors are more recent and very promising, as they have a low computational cost and can still be very descriptive. The first one is Binary Robust Independant Elementary Features (BRIEF) [12] and afterwards Oriented FAST and Rotated BRIEF (ORB) [59] and Binary Robust Invariant Scalable Keypoints (BRISK) [43] were proposed to make the descriptor invariant to rotation. Fast retina keypoint (FREAK) [1] is one of the most recently proposed method and seem to have a better descriptive power with the same computational cost as the other methods.

2.2.1 SIFT

The SIFT descriptor is a 128-dimensional real valued vector, composed of 8 float normalized histogram of gradients over a cell, with 16 cells in total. There is a mechanism to choose the best scale and the normalized orientation to be invariant over rotation and scale.

2.2.2 SURF

The SURF descriptor is a 64-dimensional real valued vector, composed of 4 float which is four sums of orientations over a cell, computed on a grid of 16 cells. The orientation is computed using a Haar wavelet filter response. It is supposed to provide invariance to illumination, viewpoint and contrast variations.

2.2.3 BRISK

BRISK is a vector of 512 bits, each bit is the short-distance binary intensity comparisons of pairs of weighted Gaussian, the position of the pair is computed using Gaussian sampling.

2.2.4 FREAK

FREAK uses the same principle as BRISK, the difference being the choice of the pairs, and the fact that the Gaussians are overlapping.

2.3 Feature Patch description

2.3.1 Haar-like

The Haar-like feature [69] is a real-valued vector, where each coordinate (float) is the result of a weighted sum over a square. There exists some extension with more features in order to account for the tilted (45 °) Haar-like features [46]. The sampling is random or in a grid at a different scale.

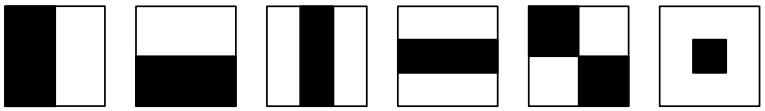


Figure 1: A standard set of Haar-like feature

2.3.2 Local Binary Patterns

The LBP feature [54] is a 8-bit value, where each bit represent the sign of a the difference between a pixel and its neighborhood. The feature extracts information from a 3x3 square. The sampling is one feature per pixel.

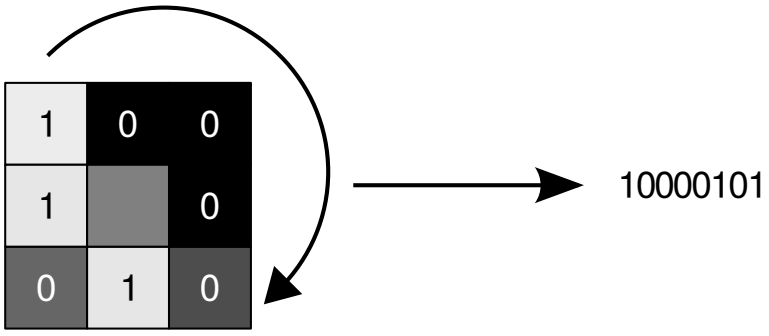


Figure 2: Creation of a LBP

2.3.3 Multi-Block LBP

The MB-LBP feature [81] is a 8-bit value. It is an extension of LBP, and the difference is that the comparison is not simply between two pixels, but between the sum of two rectangles. The feature size is over a $(3n) \times (3m)$ rectangle. The sampling cannot be exhaustive, because the number of features quickly becomes huge (166464 features for a sub-window of size 50×50). There are three ways of sampling : over a grid, at random or with feature selection.

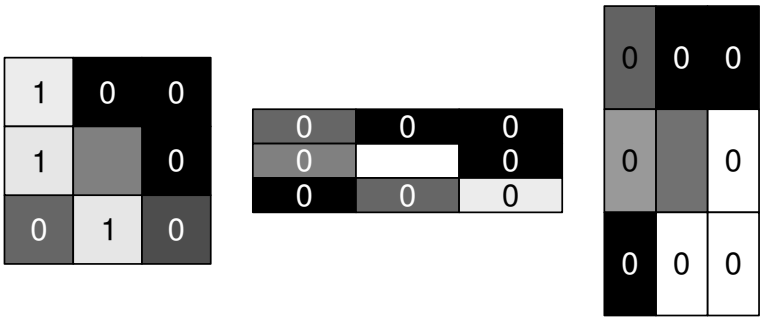


Figure 3: Example of MB-LBP

3 Machine Learning

A very common way, in recent tracking by detection approaches, is to use a variant of online boosting-based classifiers [5, 23, 60]. This yields decent result especially for particular tasks like face detection [70].
In recent object detection researches a new technique gives very promising result. It uses SVM due to its good generalization ability, robustness to label noise, and flexibility in object representation through the use of kernels [11, 21, 68].

3.1 Purpose

In tracking by detection we need to determine which sample has the biggest probability to be the object. To do that we want to use a classifier to distinguish the target object from its surrounding background. In online tracking we have only the data from the past and to save computational cost and to be as fast as possible, we want the classifier to be trained online.

3.2 Support Vector Machine

Support Vector Machines (SVM) have become very standard algorithms for classification and regression problems. They belong to the class of sparse kernel machines and are maximum margin classifiers, meaning that they minimize the classification error and maximize the geometric margin at the same time.

the SVM was initially [17] a binary decision machine. The method was first proposed as a linear classifier between two classes and cannot provide posterior probabilities, but was then extended to handle non-linearly separable problems by Kernelization.

As other so-called kernel methods, SVM handle non-lineraly separable problems, by projecting points in a higher dimensional space using a mapping. The goal is to find a projecting function to have a linearly separable problem in the new space.

Finding this function can be very difficult. To solve this issue, kernel methods rely on a principle called the kernel trick. It uses the observation that the only operation that we want to perform in the high-dimensional space is computing inner-products. So we only need to find a function allowing to compute inner-products in this space without defining the mapping explicitly. This function $k(x, x')$ is called the kernel function and must respect a few properties.
A kernel need to respect Mercer's condition. That is,

$$\iint k(x, y)g(x)g(y)\,dxdy \geq 0. \tag{1}$$

for all square integrable functions $g(x)$.
The most used kernel is :

- Gaussian Radial Basis Function kernels : $k(\mathbf{x_1}, \mathbf{x_2}) = e^{(-\gamma\|\mathbf{x_1}-\mathbf{x_2}\|^2)}$, with $\gamma > 0$.

3.3 Structured output SVM

The structured Support Vector Machine [66] is a generalization of the SVM classifier. The structured SVM allows to train of a classifier for general structured output labels.

Training the classifier consists of showing pairs of correct sample and output label pairs. After training, the structured SVM model allows one to predict for new sample instances the corresponding output label.

Training

For a set of ℓ training instances $(\boldsymbol{x}_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ from a sample space $\mathcal{X}$ and label space $\mathcal{Y}$, the structured SVM minimizes the following regularized function.

$$\min_{\boldsymbol{w}} \quad \|\boldsymbol{w}\|^2 + C \sum \max_{y \in \mathcal{Y}} \left(\Delta(y_n, y) + \langle \boldsymbol{w}, \Psi(\boldsymbol{x}_n, y) - \Psi(\boldsymbol{x}_n, y_n) \rangle\right) \tag{2}$$

The function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ measures a distance in label space. The function $\Psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ is a feature function, extracting some feature vector from a given sample and label.

Because the regularized risk function above is non-differentiable. We reformulate in terms of a quadratic program by introducing one slack variable ξ_n for each sample, each representing the value of the maximum. The standard structured SVM primal formulation is given as follows.

$$\begin{aligned} \min_{\boldsymbol{w}, \boldsymbol{\xi}} \quad & \|\boldsymbol{w}\|^2 + C \sum \xi_n \\ \text{s.t.} \quad & \langle \boldsymbol{w}, \Psi(\boldsymbol{x}_n, y_n) - \Psi(\boldsymbol{x}_n, y) \rangle + \xi_n \geq \Delta(y_n, y), \qquad \forall y \in \mathcal{Y} \end{aligned} \tag{3}$$

Inference

At test time, only a sample $\boldsymbol{x} \in \mathcal{X}$ is known, and a prediction function f: $\mathcal{X} \to \mathcal{Y}$ maps it to a predicted label from the label space $\mathcal{Y}$. For structured SVMs, given the vector $\boldsymbol{w}$ obtained from training, the prediction function is the following.

$$f(\boldsymbol{x}) = \operatorname*{argmax}_{y \in \mathcal{Y}} \quad \langle \boldsymbol{w}, \Psi(\boldsymbol{x}, y) \rangle \tag{4}$$

Therefore, the maximum y over the label space is the predicted label. Solving for the maximizer can become a hard problem depending on the structure of the function Ψ.

4 Benchmarking

Benchmarking in tracking can seem easy, because it is easy to find a metric for evaluation. Indeed We can build a video dataset and ask someone to locate a bounding box around the object for all frames. If we are careful we can ask different people to label the dataset and compare their performance to the algorithm. We just need a metric such as the Euclidean distance between the center of the box.

This technique is used by the majority of researchers to test their algorithms, and when you read the actual research the state-of-the-art seem to achieve very good results. In fact, we really need a standard framework for evaluating algorithm to compare the results between different papers.

We do not presume of the honesty of the researchers concerned, but we noticed that in general the datasets used for evaluation are really small (i.e. between three and ten videos). As often in computer vision, most of the algorithms have many parameters, which are generally invisible and hard coded. And all these parameters are optimized to provide very good results over the specific dataset used for evaluation. This is a kind of overfitting performed by the people optimizing the parameters during the implementation of the algorithm. To avoid this problem, it is important to use a sufficiently big (and diverse) dataset, with the same parameters for all the video sequences and a good evaluation method. That is why we choose to use the benchmarking method of Wu and al.[74].

4.1 Dataset

As we said it is important to use many videos comprising different kind of challenges. To understand the strengths and the weaknesses of an algorithm it is essential to organise videos in different categories according to the kind of difficulty.

Wu and al.[74] have built a dataset of 51 videos that come from different datasets. These 51 videos are categorized into 11 categories. The dataset is probably still not big enough for a completely objective evaluation, but it is a good start, and it has the advantage of allowing a comparison with the results of many state-of-the-art algorithms.

Attr	Description
IV	Illumination Variation - the illumination in the target region is significantly changed.
SV	Scale Variation - the ratio of the bounding boxes of the first frame and the current frame is out of the range $[1/2, 2]$.
OCC	Occlusion - the target is partially or fully occluded.
DEF	Deformation - non-rigid object deformation.
MB	Motion Blur - the target region is blurred due to the motion of target or camera.
FM	Fast Motion - the motion of the ground truth is larger than 20 pixels
IPR	In-Plane Rotation - the target rotates in the image plane.
OPR	Out-of-Plane Rotation - the target rotates out of the image plane.
OV	Out-of-View - some portion of the target leaves the view.
BC	Background Clutters - the background near the target has the similar color or texture as the target.
LR	Low Resolution - the number of pixels inside the groundtruth bounding box is less than 400 pixels.

Figure 4: The categories used by Wu and al.

Figure 5: Samples of the video sequences constituting the dataset

4.2 Evaluation Methodology

A classical way to have a quantitative evaluation is to compute the average Euclidean distance between the center of the ground truths and the center of the tracked box. The main issue is that when the tracker looses the target then this distance becomes totally random. They do not propose a real solution to avoid this problem. The proposed solution is to run the tracker several times with different starting frames or different starting bounding boxes.

4.2.1 Precision plot

The precision plot was proposed recently in [6, 30]. It shows the percentage of frames for which the distance between the tracked target and the ground truth is below a threshold. The precision plot is the precision rate in function of the threshold.

4.2.2 Success plot

Another evaluation metric is the bounding box overlap. The overlap is simply defined as the intersection area divided by the union area of the tracked box and the ground truth box. We use the same technique as the precision plot to build the success plot.

4.2.3 Ranking

In order to establish a ranking between different algorithms we can use the Area Under the Curve (AUC) of the success or precision plot. There is no clear evidence that this metric is the most objective one, but this is a good insight of the real performance.

4.2.4 Robustness Evaluation

To avoid the problem of the sensitivity at the initialization we use two techniques.

1. For each video we evaluate the algorithm several times with different starting frames.
2. For each video we evaluate the algorithm several times with different starting bounding boxes.

Part II
Contributions and Applications

5 Motivation

The motivation to start this master thesis was to study and try to improve the state-of-the-art of visual tracking, and the obvious first step is to evaluate it. To do that we use the study of Wu and al. [74], as they compare many different algorithms using more video sequences for the tests than usual. They also provide a good benchmarking method, which is really important for visual tracking as explained before.

One algorithm stands out : Struck [27] which has a great performance and is realtime (25 fps in average). It is a very interesting work because the mathematical formulation is clean and elegant.

The main innovation comes from the machine learning side. On the other hand, the description of the features is very classical, and our first contribution is to improve this part. We will present the machine learning part of Struck because it is very informative, but we do not improve it.

5.1 State of the art

The standard technique used for traditional tracking by detection algorithms is to learn a classifier in order to separate the target from the background.

The classifier is trained with example pairs $(\mathbf{x}, z)$ where $\mathbf{x}$ is the feature vector of a patch and $z = \pm 1$ its corresponding binary label. The function $f : \mathcal{X} \to \mathbb{R}$ being the classification confidence function, the prediction corresponds to $\hat{z} = sign(f(\mathbf{x}))$.

To estimate the new position, we search the maximum of $f(\mathbf{x})$ around the initial estimated position (in general we use the position of the previous frame). Let p_{t-1} be the estimated bounding box (position and size) at time $t-1$. The goal of the tracker is to estimate a transformation[1] $y_t \in \mathcal{Y}$ such that the new position (or the new state) of the bounding box is estimated by the composition $p_t = p_{t-1} \circ y_t$.Here, $\mathcal{Y}$ denotes the search space and its form depends on the type of motion to be tracked (generally a translation, but in this work we extend the search space of Struck to integrate scale transformations.

Finally we obtain:

$$p_t = p_{t-1} \circ y_t \tag{5}$$

$$y_t = \arg\max_{y \in \mathcal{Y}} f(\mathbf{x}_t^{p_{t-1} \circ y}) \tag{6}$$

where $\mathbf{x}_t^{p_{t-1} \circ y}$ are the feature vectors extracted from the frame at time t with the bounding box $p_{t-1} \circ y$. When we have the new estimate for the bounding box, we can build a set of training data from the current frame. It is divided into two components: the sampler and the labeller.

The sampler generates a set of transformations $\{y_t^1, ..., y_t^n\}$ with a corresponding set of feature vectors $\{\mathbf{x}_t^{p_t \circ y_t^1}, ..., \mathbf{x}_t^{p_t \circ y_t^n}\}$ and the labeller chosen for these training set $\{z_t^1, ..., z_t^n\}$. Finally, using this information, we can update the classifier.

In the work of Hare and al. [27] the authors raise some problems that can arise with this technique (and Struck try to avoid these problems).

- The assumption made in (6) that the classification confidence function provides an accurate estimate of the object position is not explicitly incorporated into the learning algorithm. Why hiding the information about the transformation to the classifier?

[1] Generally a translation, but it can be a rotation or zoom

- The examples used to train the classifier are all equally weighted, meaning that a negative example which overlaps significantly with the tracker bounding box is treated the same as one which overlaps very little. We thus loose a very important part of the information.
- The labeller is usually chosen based on intuitions and heuristics, rather than having a tight coupling with the classifier. Mistakes made by the labeller manifest themselves as label noise.

Many current state-of-the-art approaches try to overcome these problems by using robust loss functions [42, 49], semi-supervised learning [24, 61], or multiple-instance learning [5, 80].

All these methods are essentially workarounds for the real problem which stems from the fundamental issue of separating the labeller from the learner. Struck does not depend on a labeller, and tries to overcome all these problems by using a structured SVM to learn the transformation function.

5.2 Struck

A transformation function

$$f : \mathcal{X} \to \mathcal{Y} \tag{7}$$

is used to estimate the object transformation between frames, where $\mathcal{X}$ is the space of the feature vector and $\mathcal{Y}$ is the space of all transformations. Note that it avoids the use of binary labels $z = \pm 1$. A labelled example becomes a pair $(\mathbf{x}, y)$. The function f is learned using a structured-output SVM framework [11, 66], which introduces a discriminant function

$$F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \tag{8}$$

that we use for prediction according to

$$y_t = \arg\max_{y \in \mathcal{Y}} F(\mathbf{x}_t^{p_{t-1}}, y) = f(\mathbf{x}_t^{p_{t-1}}) \tag{9}$$

The difference with equation (6) is that now F is a function of y so it can be incorporated in the learning algorithm.
F measures the compatibility between $(\mathbf{x}, y)$ pairs by restricting this to be of the form

$$F(\mathbf{x}, y) = \langle \boldsymbol{w}; \Psi(\mathbf{x}, y) \rangle \tag{10}$$

where $\Psi(\mathbf{x}, y)$ is a joint kernel map. And now we can use the structured SVM to learn from a set of example pairs $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$ by minimizing the convex objective function:

$$\begin{array}{ll} \min\limits_{\boldsymbol{w}, \boldsymbol{\xi}} & \|\boldsymbol{w}\|^2 + C \sum_{i=0}^{n} \xi_i \\ \text{s.t.} & \langle \boldsymbol{w}, \Psi(\boldsymbol{x}_i, y_i) - \Psi(\boldsymbol{x}_i, y) \rangle + \xi_i \geq \Delta(y_i, y), \qquad \forall y \in \mathcal{Y} \end{array} \tag{11}$$

This optimization ensures that the value of $F(\mathbf{x}_i, y_i)$ is greater than $F(\mathbf{x}_i, y)$ for $y \neq y_i$, by a margin which depends on a loss function Δ. This loss function should satisfy the following properties :

- $\Delta(y, y_i) = 0 \iff y = y_i$
- Δ decreases towards 0 as the similarity between y and y_i increases

The loss function plays an important role in this method, as it allows us to address the issue raised previously of all samples being treated equally. In the original Struck, they propose to use the bounding box overlap:

$$\Delta(y, y_i) = 1 - a(y, y_i) \tag{12}$$

Where a is the overlap ratio. In order to improve Struck we tried several other functions[2] but the overlap function (12) achieved the best results.

[2]We tried to build a function based on the distance of the two bounding boxes with or without normalization.

6 Feature Choice

6.1 Motivation

To build a good algorithm for visual tracking, we need a fast and smart technique to describe a patch. Here we try to define what is a good description.

The description is obviously dependent of the algorithm, and so we do not aim at preserving the same properties for all algorithms.

We want to approximate the distance given by a human, with the constraint of the visual tracking. To approximate this we need to care about some properties.

6.1.1 Realtime constraint

It may seem strange to mention this property first, but if the algorithm cannot be run in realtime, why bother using an online algorithm as we could use the information from all the frames from the sequence instead. It is interesting to explore different ways, but for a usable algorithm, we need to restrict ourselves only to the class of realtime algorithms.

If we feed Struck with raw values only, using the appropriate scale, we see that the performance (i.e. measured by the quality of the tracking) remains acceptable. The speed constraint is really important and limits the fields of usable descriptors.

6.1.2 Illumination invariance

It is often one of the first property that we require for a good descriptor. Haar-like features, LBP, and most of keypoint descriptors are resistant to illumination variations. Generally if the descriptor is not illumination invariant, it is so by design (e.g. histograms).

6.1.3 Scale invariance

Scale invariance is a nice-to-have but not necessary property since it can be de-localised to other components. For example in our method we added scale adaptation without having this property built in the feature descriptor. In general this is not really important because the scale changes slowly.

6.1.4 Rotation invariance

Rotation invariance is nice to have but not necessary for the same reason as the scale, i.e. the rotation varies slowly so it can be incorporated in the model adaptation. Most feature points are rotation invariant, but it is harder to make a patch descriptor resistant in the same way. The tricks used for the scale is unusable or inefficient due to use of an integral image to compute the descriptor.

6.2 Binary keypoint descriptor

Due to the recent success of binary descriptors such as FREAK [1]. We tried to use it for the tracking, with a naive approach. We simply put one or several keypoints in the patch. The result is good given the quantity of information (64 bytes for one FREAK descriptor), but the performance is significantly worse than the Haar-like features from the original Struck paper.

The possible explanation is that the time saved during the comparison is largely lost during the extraction of the keypoint. This is not the use for which keypoint

descriptors have been designed. Normally the number of comparisons between descriptors is significantly bigger than the number of descriptors to extract.

In the end we want to keep the principle of the binary representation, but with faster extraction and with a better descriptive power for a patch. The LBP appears like an evidence for the seeked properties.

6.3 MB-LBP

We propose to use the MB-LBP features, due to their excellent results in the field of face recognition and this is exactly the generalization for a patch of the technique of FREAK, BRISK and BRIEF. The main problem of a keypoint descriptor is that it is centered around a point, whereas in a patch nothing say that the interesting information lies only at the center. Since MB-LBP features can be positioned anywhere in the patch, then the extracted information is not related to one central point.

The problem is the number of possible MB-LBP. A patch of 50x50 pixels in 8 bits gray level is 2500 bytes. In the same patch, it is possible to extract 166464 different MB-LBP features, where one MB-LBP is one byte. We thus have 65 times more data with all the MB-LBP than using the pixel information.

Taking all the MB-LBP is impossible (due to the computational cost) and if we look at the work of Paratte [55], this seems like a bad idea [3].

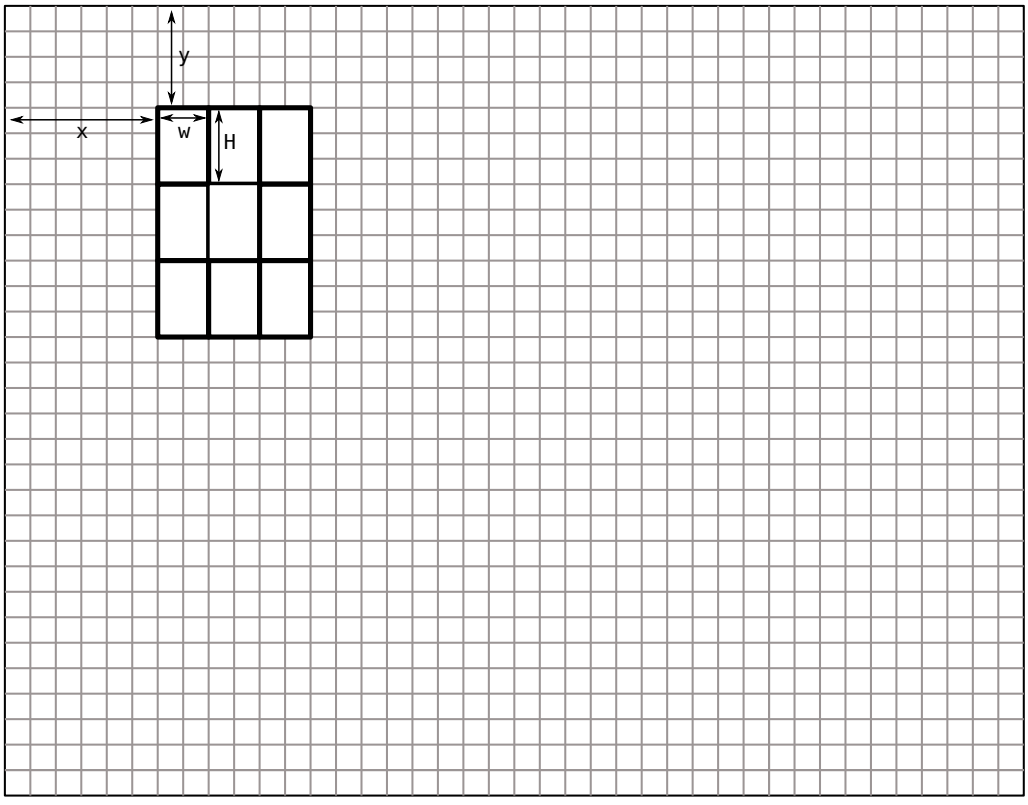


Figure 6: Representation of an MB-LBP with (x, y, w, h).

[3] for FREAK taking all the pairs is worse than a well chosen subset of pairs

6.3.1 Fully random sampling

To start we choose the MB-LBP randomly, without using any heuristics. First, we choose the size randomly and then we choose the position also at random. The probability is uniform over the size, but obviously not over the MB-LBP, since there are more small MB-LBP than big ones. We show an histogram of 1600 MB-LBP generated with this technique (the number 1600 is chosen in order to have approximately the same number of bytes than the Haar-like representation chosen in Struck).

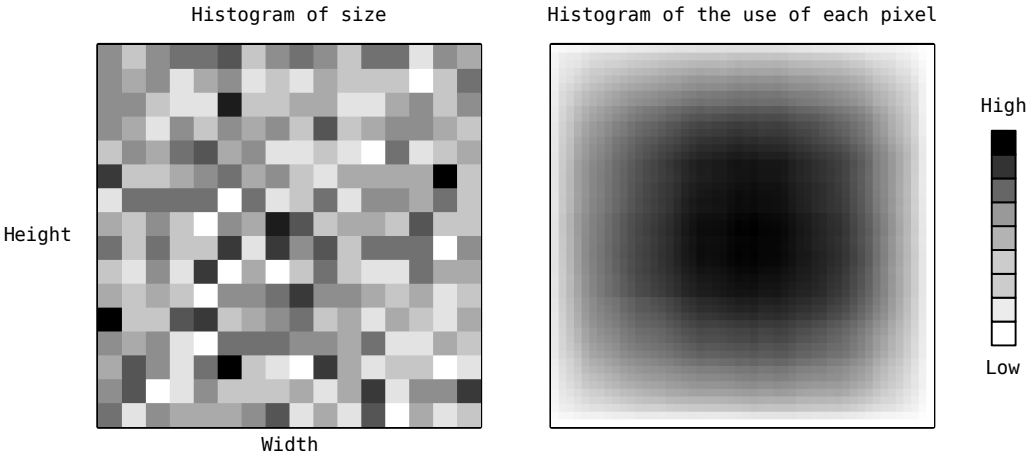


Figure 7: Histogram of random MB-LBP with uniformly sampled size.

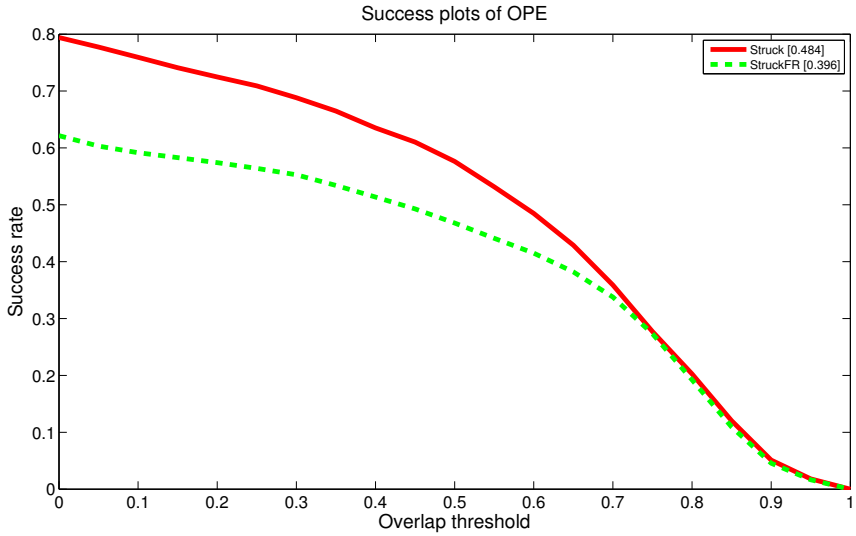


Figure 8: Struck is with Haar-like feature. StruckFR is with MB-LBP uniformly random on the size.

In figure 8 we can see the success plot corresponding to the proposed method using randomly chosen MB-LBP, sadly the result as not as good as expected. This shows that if we choose the set of features too naively, the result is very poor.

6.3.2 Feature selection

Introduction

We want to find a good subset of features. The best subset of features for this algorithm is the set that achieves the best performance. We can define a score that we want to maximize, e.g. the AUC in the overlap plot defined by Wu and al.. We also need a dataset, so we could try to use a relatively small data set, but if we want not to overfit the data, we really need to choose a data set sufficiently big. The problem of feature selection is NP-Hard so we need to find heuristics to help us.

The goal is to separate the object from the background. Thus we want to have a small distance when we compare patches of the object over time, and have a big distance when we compare patches of the object and patches coming from the background.

Idea

First we use the ground truth to extract patches from the object and from the background. Then we compute the distance of a feature from the object on the frame t and frame $t-l$ with l small. This distance is called distance of true match, and the distance between the object and the background is called distance of false match. A good feature is a feature with small mean over time of the distance of true match and a big mean over the time of the distance of false match.

Sampling

We start by fixing the size of the patch (in our case we choose 50×50 pixels and we will show that it is sufficient), for all other patch sizes we use a projection. It is an idea for the future work to generate several different patch sizes with different ratios, and finally matching with the closest patch size (in the sens of the ratio).

Now, we use an heuristic to reduce the number of features. For each size we build a grid for the sampling, with an horizontal and vertical spacing equal to the width and the height respectively. This means that two MB-LBP with same size that are side by side overlap for two third. We keep only the MB-LBP inside the patch and we center the grid.

With this heuristic, the number of feature is reduced from 166464 to 17424.

Selection algorithm

For the selection part we use an algorithm developed by Paratte during his master thesis [55]. Based on his work, we conclude that selecting good features (as defined above) is not enough, we need to select features with low correlation. His work on selection of FREAK pairs is really similar to this feature selection.

1. For each frame we extract one patch from the object, and 8 patches from the background around the object, to find the object we use the ground truth.
2. We extract the set of 17424 features for each patch.
3. For each feature we compute the mean distance between the object at frame t up to $t-l$ ($l = 2$ yields good result), this gives the true matching score.
4. For each feature we compute the mean distance between the object at frame t and with the background at the same frame. We subtract this mean at 8 (it is the maximum distance between two features [4]) to obtain the false matching score.

[4] this the Hamming distance between two bytes

5. We compute a total score for each feature, with a linear combination between the true and the false matching scores.

6. We rank all the features according to their matching score, they form the rank set.

7. We create an empty set (the selected set).

8. We take the feature with highest matching score from the rank set, we compute the maximum correlation between this feature and all the ones from the selected set, if it is below the maximum correlation threshold then we transfer the feature from the rank set to the selected set, otherwise we delete this feature.

9. If we have the desired number of features in the selected set we stop and keep the selected set. If the rank set is empty but the number of selected feature is not reached we increase the maximum correlation threshold and restart to 6. Otherwise we continue with 8.

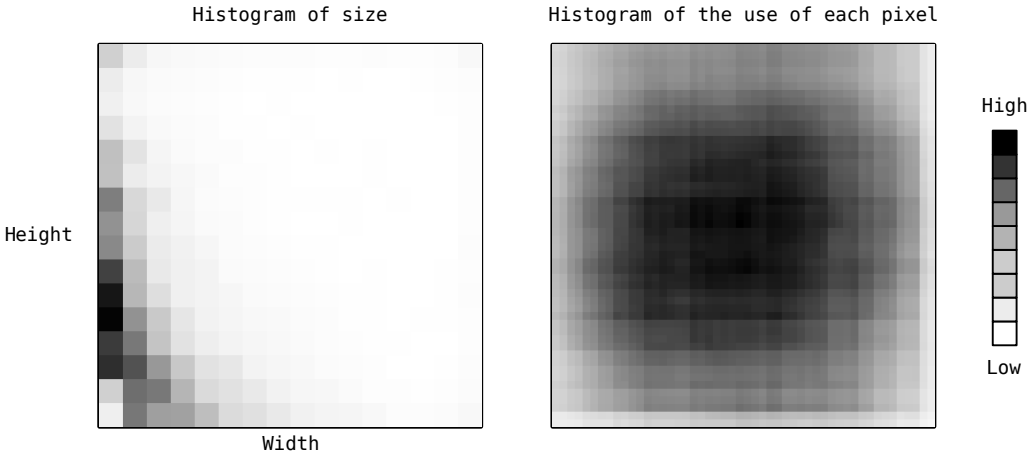


Figure 9: Histogram of selected MB-LBP.

In figure 9 we see which features are selected.

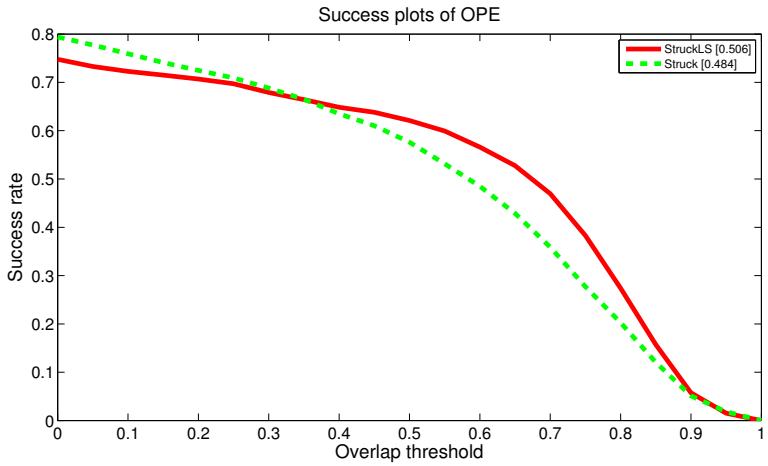


Figure 10: Struck is the original implementation with Haar-like feature. StruckLS is with the selected set of MB-LBP.

In figure 10 we see that with the same amount of data for the description, we outperform the original Struck.

7 Extension of Search Space

7.1 Size adaptation

The implementation of Struck proposed by Hare and al. cannot adapt to variation in tracked object size. We propose a simple technique to tackle this problem.

The description has to adapt to different sizes and ratio, these are the only properties that we need to have.

The MB-LBP based technique proposed works perfectly, because we project a patch of size 50×50 on the desired patch, if the desired patch change of size during the tracking, the features remains comparable.

We simply extend the search space to add the change of size of the bounding box. This works because the change of size is smooth and relatively slow (small size change).

For a bounding box at frame t (x_t, y_t, w_t, h_t) the search space of Struck is:

$$(x_{t+1}, y_{t+1}, w_{t+1}, h_{t+1}) = (x_t + i, y_t + j, w_t, h_t) \qquad i, j = \{-30, ..., 30\} \tag{13}$$

The new search space proposed is simply:

$$\begin{aligned}(x_{t+1}, y_{t+1}, w_{t+1}, h_{t+1}) = (x_t + i, y_t + j, w_t + k, h_t + l)\\ i, j = \{-30, ..., 30\} \quad k, l = \{-1, 0, 1\}\end{aligned}$$

All the benchmarks are built with this modification and the performance improvement is noticeable but not so big. This is probably due the fact that when the tracker looses the object position, the change of size becomes very random.

7.2 Rotation adaptation

We can use the same approach, to add some resistance to orientation variations. The hypothesis of small change is still valid. The problem comes from the actual implementation of the algorithm. Indeed, in order to save a lot of computational power we use integral images. The proposed solution is to compute a rotated image with an angle ($+10°$ and $-10°$), and then to compute the integral image of these two images.

We sample as described above on the three images $(+10°, 0, -10°)$.

Due to the limited time this implementation has not been tested.

8 Context-awareness (tracking)

We propose a new way to take the context into account, based on the feature selection.

8.1 Idea

We use the feature selection technique to choose some features in the context to improve the tracking process, the technique is simple but the results validate the approach.

Figure 11: In green the tracked bounding box, in blue the up scaled bounding box.

8.2 Algorithm

For the feature selection step we up-scale the ground truth of a scale factor (generally between $1.2 - 1.5$). We select the features with the same technique as describe.

We then track the object with a bounding box and extract the features with a up scaled bounding box.

We assume that the selection algorithm chooses the good features. The method is thus strengthened, as shown by the results. Without the selection part of the feature, if we use Haar-like features as the classical Struck, the best result is achieved with a scale factor of 1.0, so without any scaling.

But with the selection process the best result is with a scale between 1.2 and 1.5.

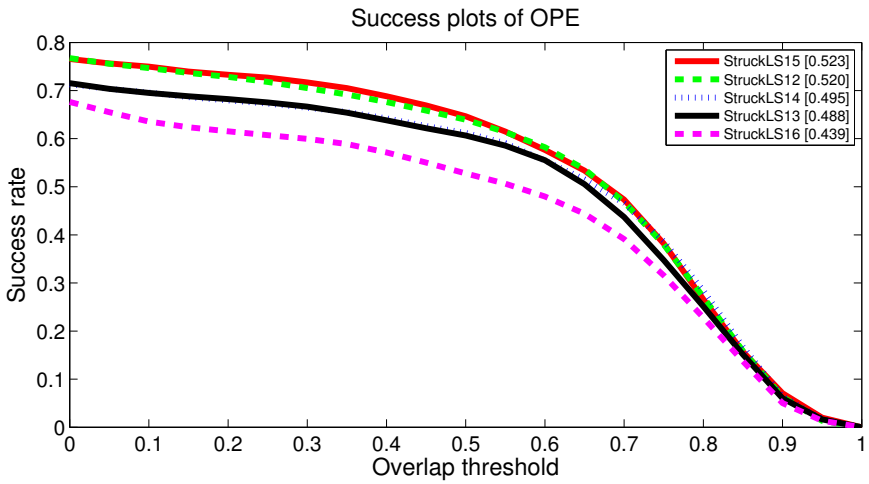


Figure 12: Run at different scales.

In figure 12 The result is a little bit strange because the result in function of the scale does not look convex. We do not have enough time investigate deeper.

9 Benchmarking

To evaluate the performance, we use the technique proposed by Wu and al.. We do not modify the framework to keep the neutrality as explained above. We show plots, one by category and one for the overall performance. We display the two evaluation metrics (overlap and distance).

We use the temporal robustness evaluation, this means that for each video we run 20 times the algorithm with different starting positions.

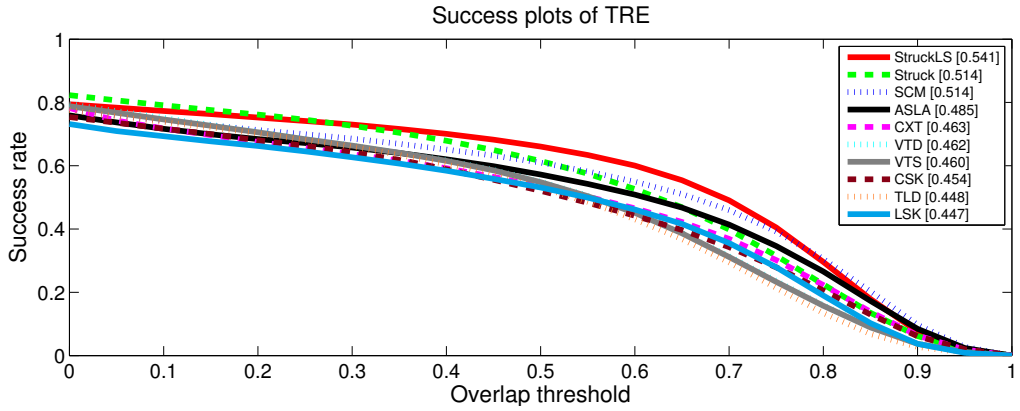
To achieve this performance we use all the ameliorations presented in this document, i.e. feature selection, size adjustment and context awareness. The parameters are set one time for all the video, so there is no specific parameter fitting for each video. We name the algorithm StruckLS.

In the following Figures are displayed the nine best state-of-the-art algorithms plus StruckLS, so that we can compare the overall performance.

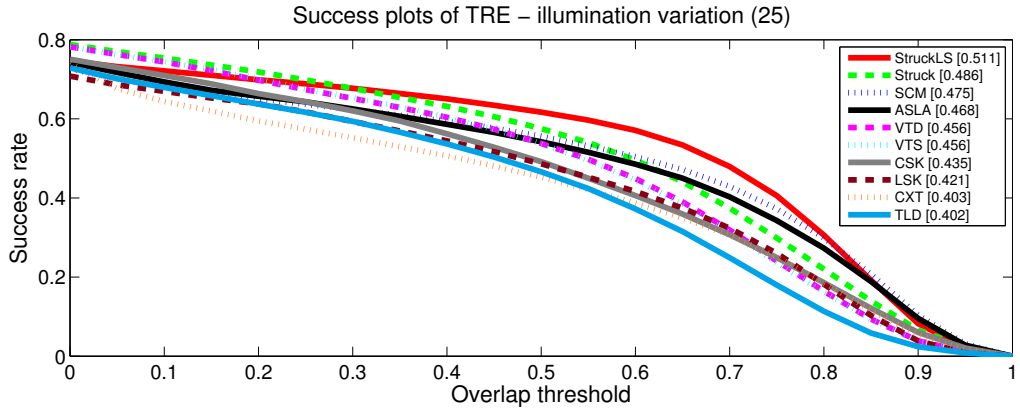
Note that the color depends of the AUC score in each plot. And the number in the title of the graph is the number of videos in the category.

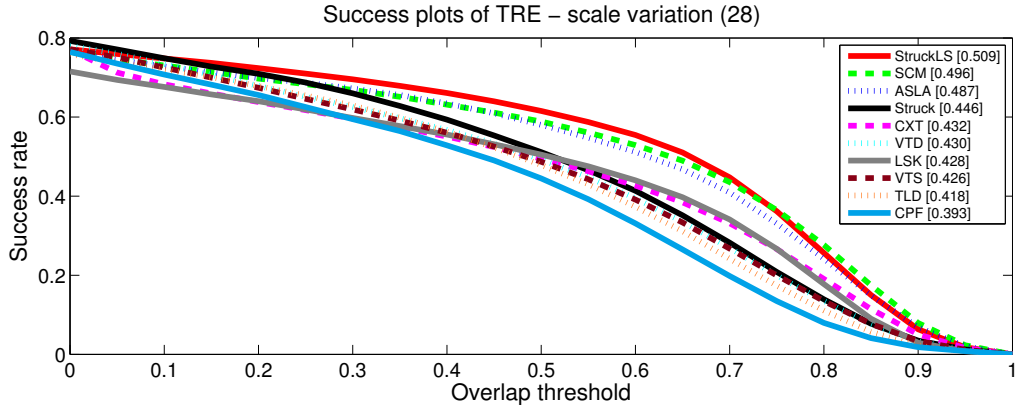
9.1 Success plot

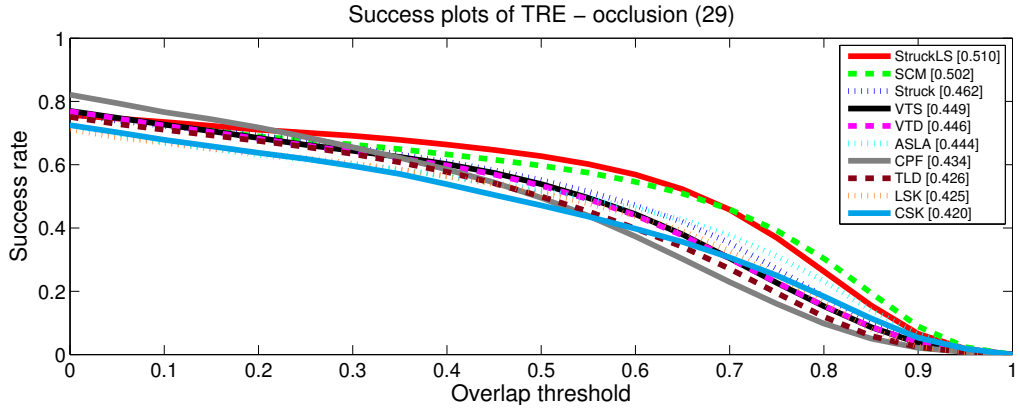
In the Figures we show first the success plots (based on the overlap ratio). It the most interesting, because the precision plot is a little bit artificial[5].

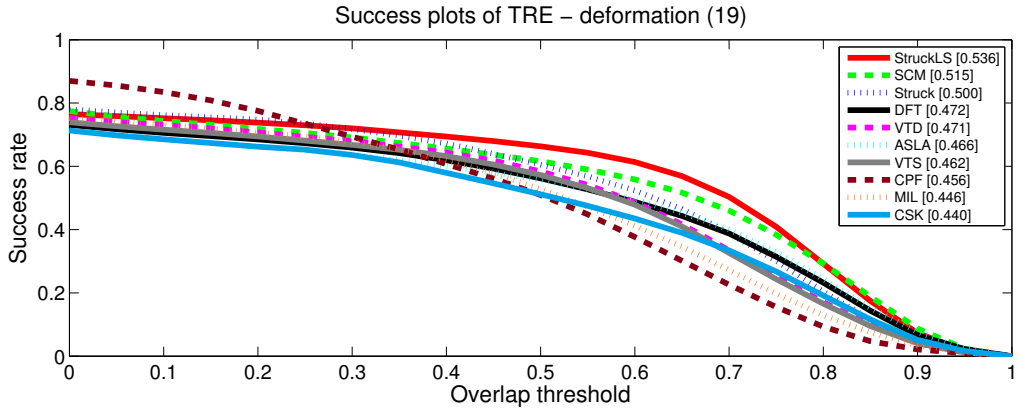


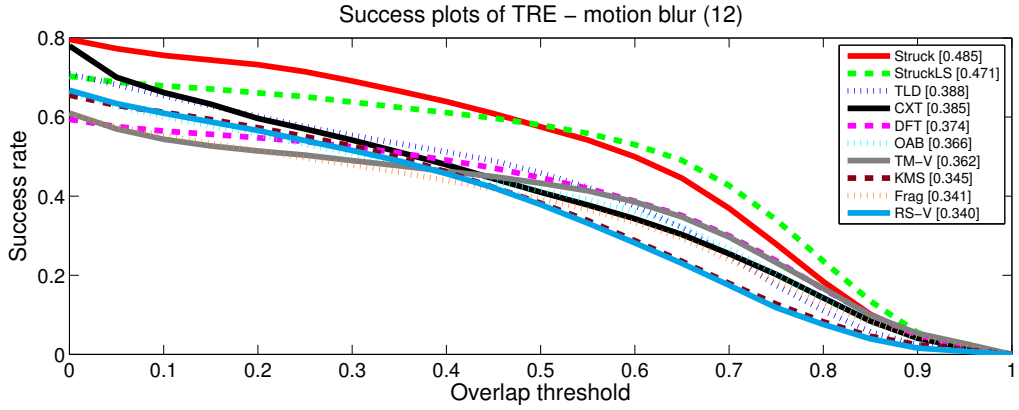
[5]Indeed the distance (in pixel) is highly dependent on the image resolution and on the bounding box size.

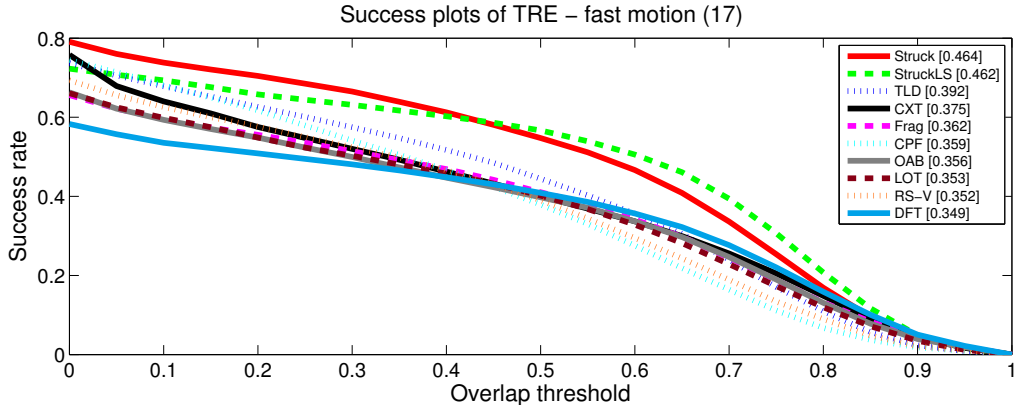

Success plots of TRE – illumination variation (25)
Success rate
Overlap threshold
StruckLS [0.511]
Struck [0.486]
SCM [0.475]
ASLA [0.468]
VTD [0.456]
VTS [0.456]
CSK [0.435]
LSK [0.421]
CXT [0.403]
TLD [0.402]

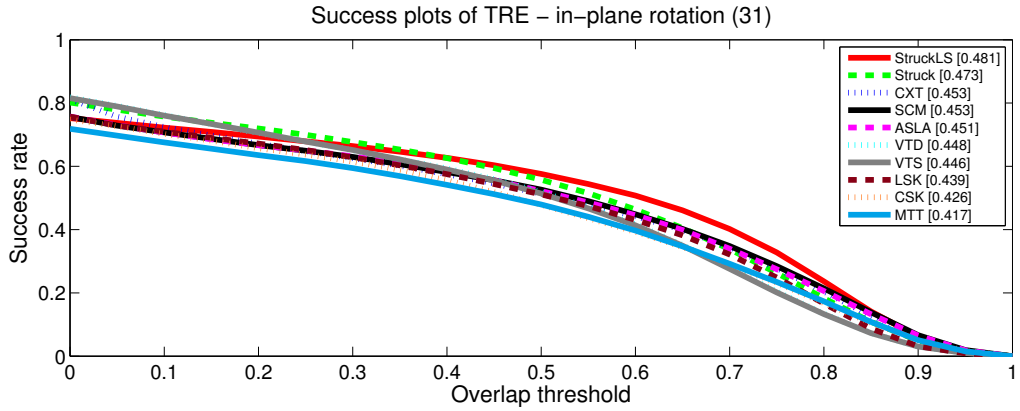

Success plots of TRE – scale variation (28)
Success rate
Overlap threshold
StruckLS [0.509]
SCM [0.496]
ASLA [0.487]
Struck [0.446]
CXT [0.432]
VTD [0.430]
LSK [0.428]
VTS [0.426]
TLD [0.418]
CPF [0.393]

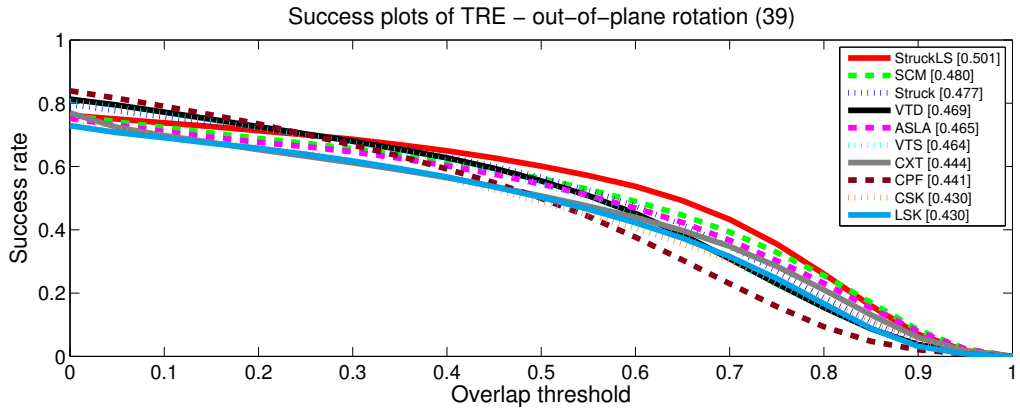

Success plots of TRE – occlusion (29)
Success rate
Overlap threshold
StruckLS [0.510]
SCM [0.502]
Struck [0.462]
VTS [0.449]
VTD [0.446]
ASLA [0.444]
CPF [0.434]
TLD [0.426]
LSK [0.425]
CSK [0.420]

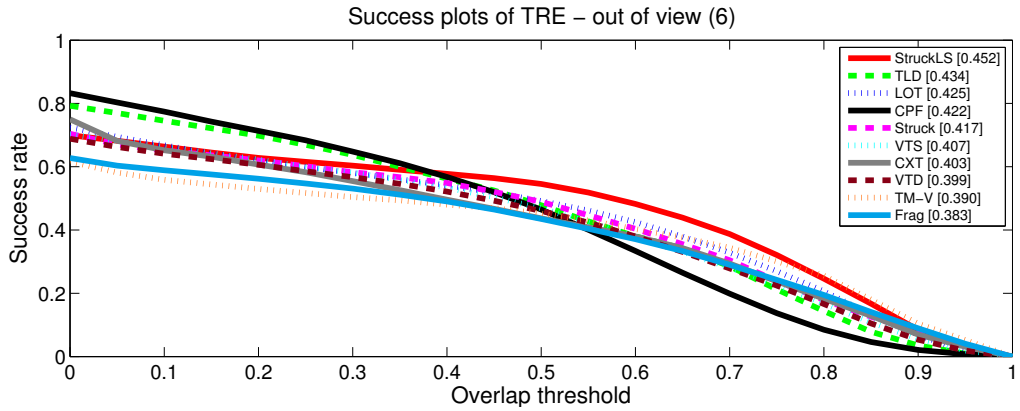

Success plots of TRE – deformation (19)
Success rate
Overlap threshold
StruckLS [0.536]
SCM [0.515]
Struck [0.500]
DFT [0.472]
VTD [0.471]
ASLA [0.466]
VTS [0.462]
CPF [0.456]
MIL [0.446]
CSK [0.440]

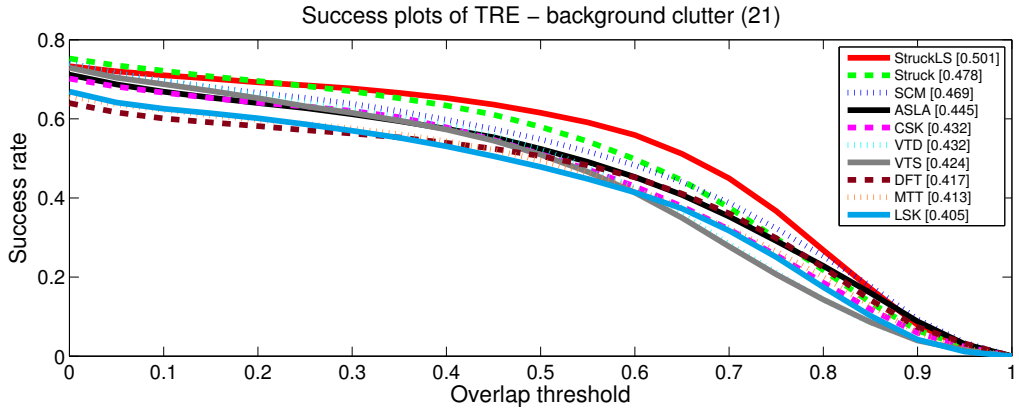

Success plots of TRE – motion blur (12)
Success rate
Overlap threshold
Struck [0.485]
StruckLS [0.471]
TLD [0.388]
CXT [0.385]
DFT [0.374]
OAB [0.366]
TM–V [0.362]
KMS [0.345]
Frag [0.341]
RS–V [0.340]

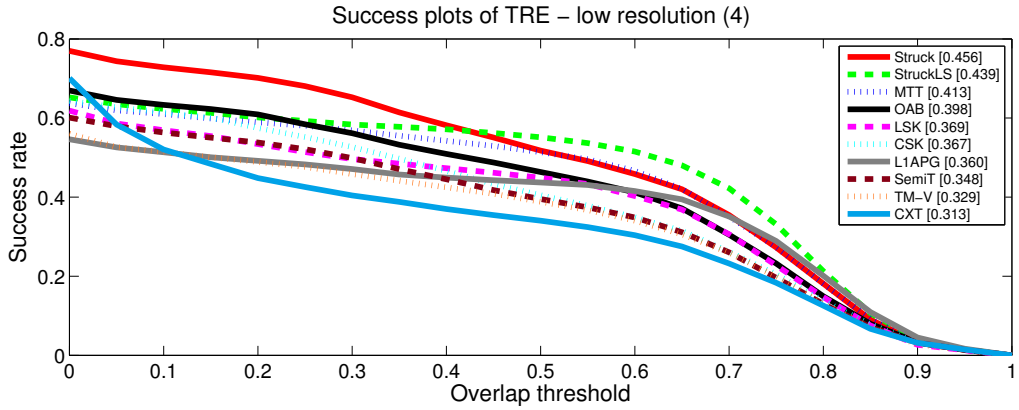

Success plots of TRE – fast motion (17)
Success rate
Overlap threshold
Struck [0.464]
StruckLS [0.462]
TLD [0.392]
CXT [0.375]
Frag [0.362]
CPF [0.359]
OAB [0.356]
LOT [0.353]
RS–V [0.352]
DFT [0.349]


Success plots of TRE – in–plane rotation (31)
Success rate
Overlap threshold
StruckLS [0.481]
Struck [0.473]
CXT [0.453]
SCM [0.453]
ASLA [0.451]
VTD [0.448]
VTS [0.446]
LSK [0.439]
CSK [0.426]
MTT [0.417]

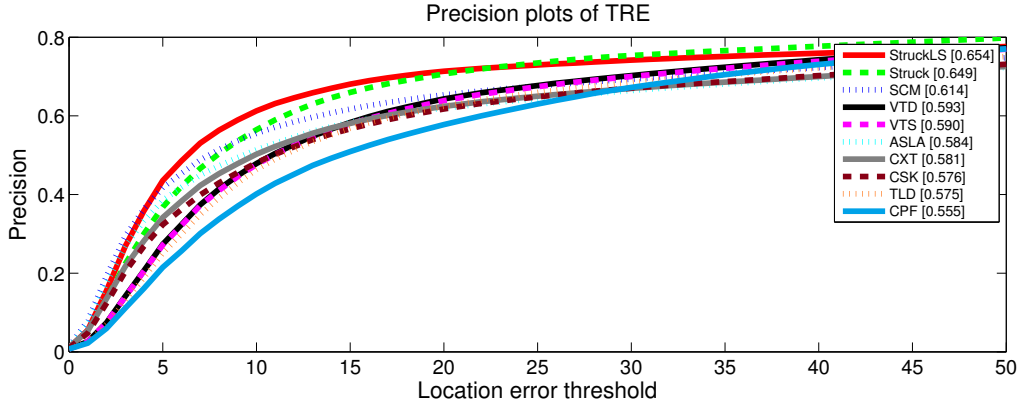

Success plots of TRE – out–of–plane rotation (39)
Success rate
Overlap threshold
StruckLS [0.501]
SCM [0.480]
Struck [0.477]
VTD [0.469]
ASLA [0.465]
VTS [0.464]
CXT [0.444]
CPF [0.441]
CSK [0.430]
LSK [0.430]

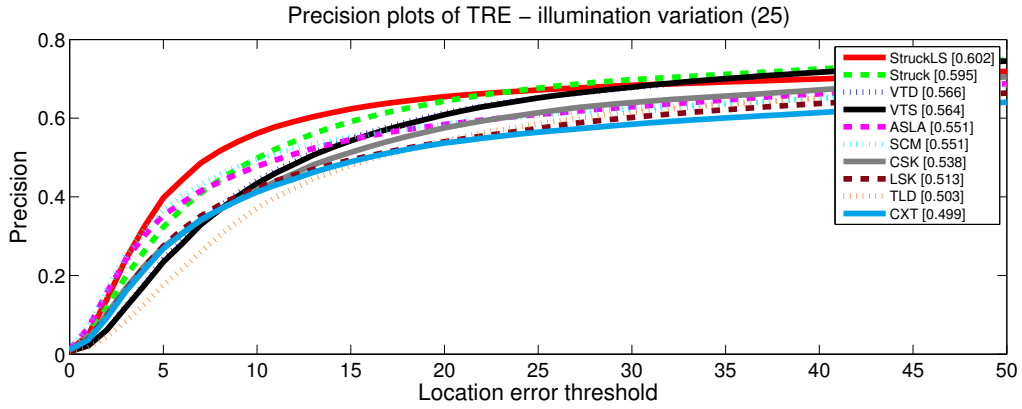

Success plots of TRE – out of view (6)
Success rate
Overlap threshold
StruckLS [0.452]
TLD [0.434]
LOT [0.425]
CPF [0.422]
Struck [0.417]
VTS [0.407]
CXT [0.403]
VTD [0.399]
TM–V [0.390]
Frag [0.383]

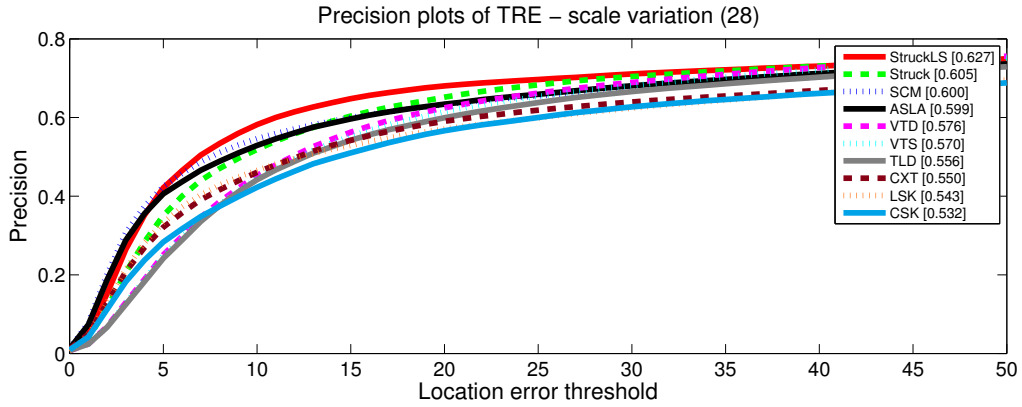


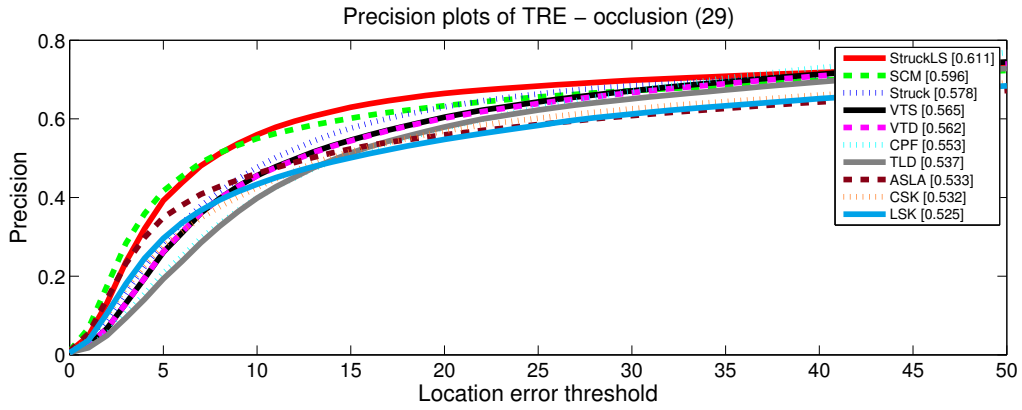


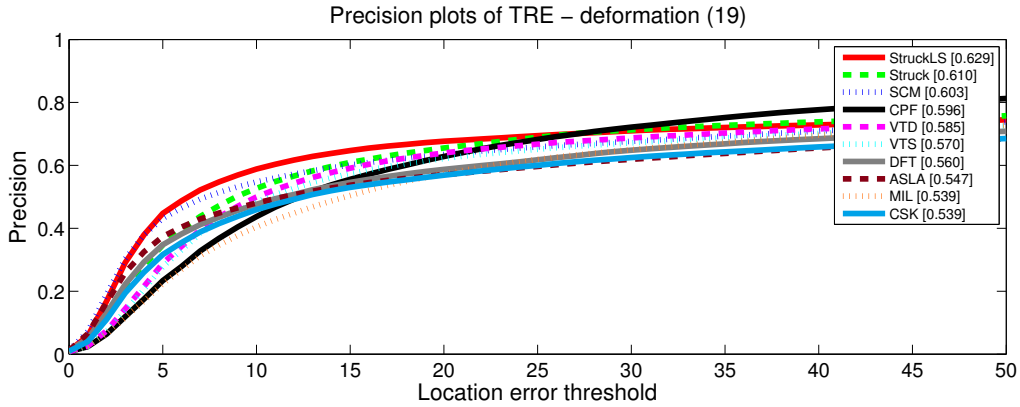
9.2 Precision plot

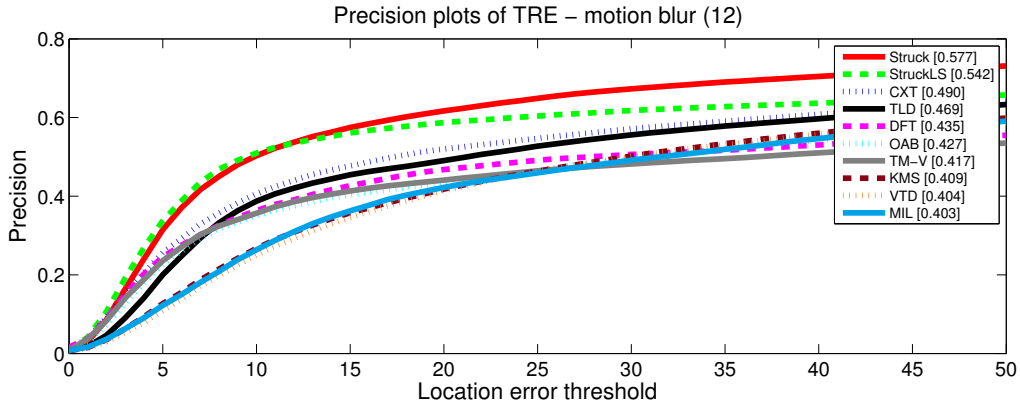


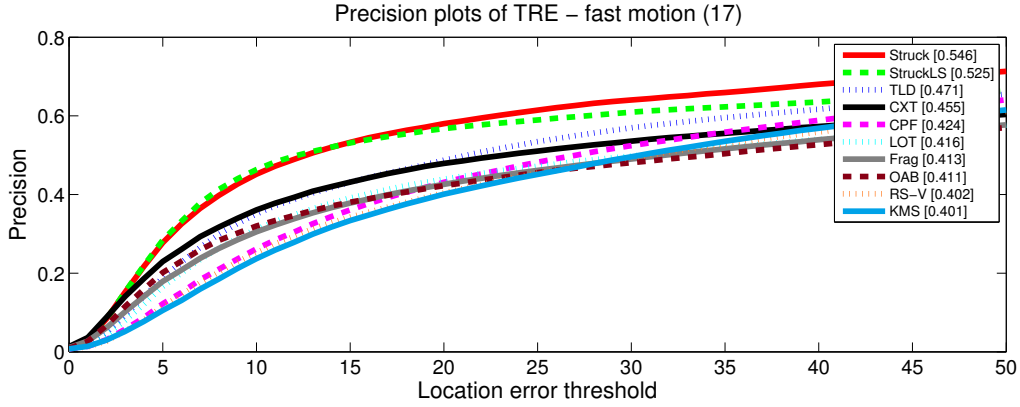

Precision plots of TRE – illumination variation (25)
Precision
Location error threshold
StruckLS [0.602]
Struck [0.595]
VTD [0.566]
VTS [0.564]
ASLA [0.551]
SCM [0.551]
CSK [0.538]
LSK [0.513]
TLD [0.503]
CXT [0.499]

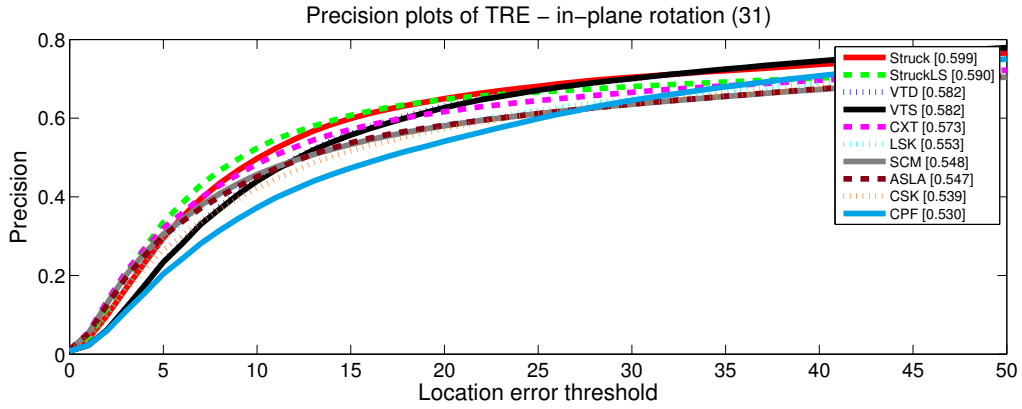

Precision plots of TRE – scale variation (28)
Precision
Location error threshold
StruckLS [0.627]
Struck [0.605]
SCM [0.600]
ASLA [0.599]
VTD [0.576]
VTS [0.570]
TLD [0.556]
CXT [0.550]
LSK [0.543]
CSK [0.532]

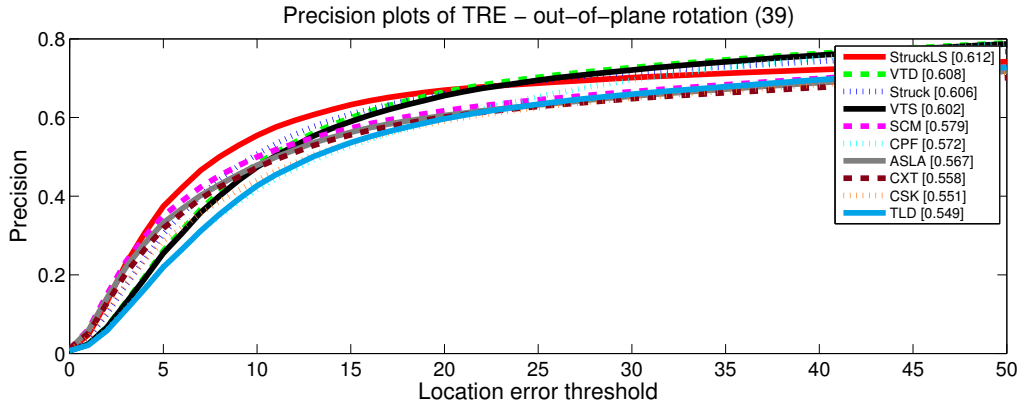

Precision plots of TRE – occlusion (29)
Precision
Location error threshold
StruckLS [0.611]
SCM [0.596]
Struck [0.578]
VTS [0.565]
VTD [0.562]
CPF [0.553]
TLD [0.537]
ASLA [0.533]
CSK [0.532]
LSK [0.525]

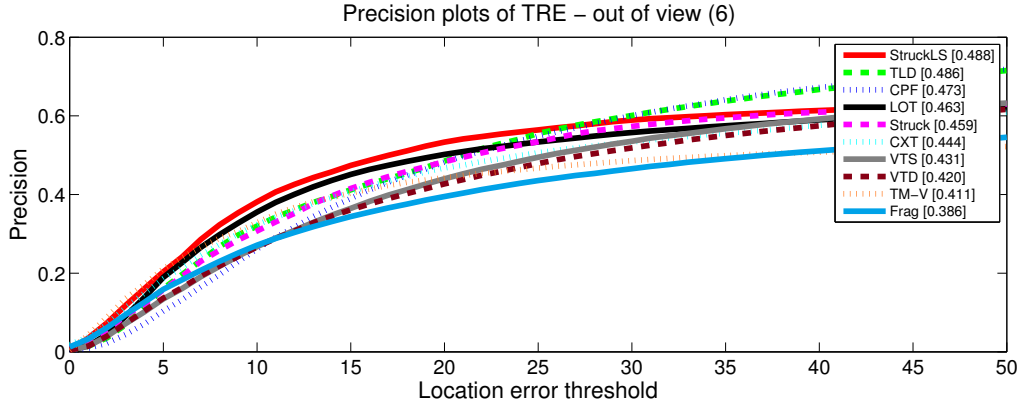

Precision plots of TRE – deformation (19)
Precision
Location error threshold
StruckLS [0.629]
Struck [0.610]
SCM [0.603]
CPF [0.596]
VTD [0.585]
VTS [0.570]
DFT [0.560]
ASLA [0.547]
MIL [0.539]
CSK [0.539]

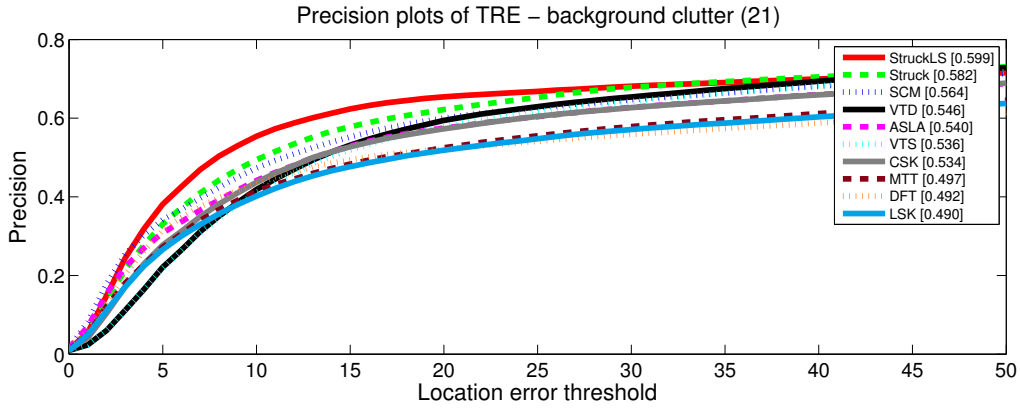

Precision plots of TRE – motion blur (12)
Precision
Location error threshold
Struck [0.577]
StruckLS [0.542]
CXT [0.490]
TLD [0.469]
DFT [0.435]
OAB [0.427]
TM–V [0.417]
KMS [0.409]
VTD [0.404]
MIL [0.403]

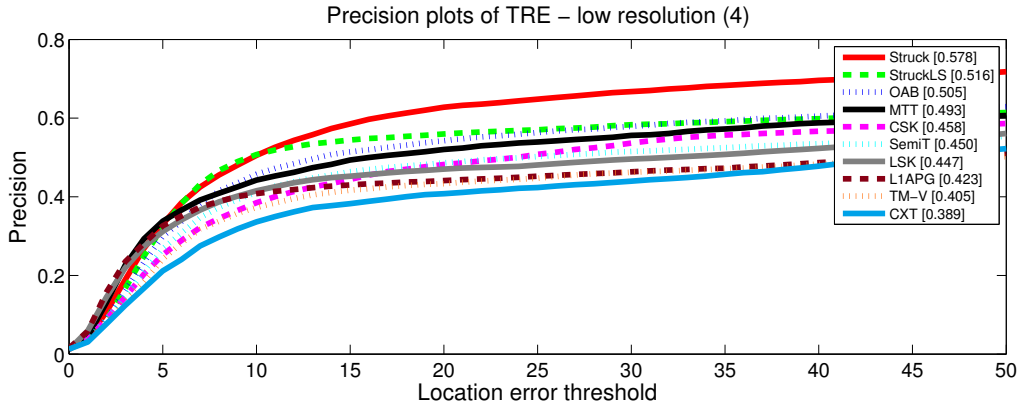

Precision plots of TRE – fast motion (17)
Precision
Location error threshold
Struck [0.546]
StruckLS [0.525]
TLD [0.471]
CXT [0.455]
CPF [0.424]
LOT [0.416]
Frag [0.413]
OAB [0.411]
RS–V [0.402]
KMS [0.401]


Precision plots of TRE – in–plane rotation (31)
Precision
Location error threshold
Struck [0.599]
StruckLS [0.590]
VTD [0.582]
VTS [0.582]
CXT [0.573]
LSK [0.553]
SCM [0.548]
ASLA [0.547]
CSK [0.539]
CPF [0.530]


Precision plots of TRE – out–of–plane rotation (39)
Precision
Location error threshold
StruckLS [0.612]
VTD [0.608]
Struck [0.606]
VTS [0.602]
SCM [0.579]
CPF [0.572]
ASLA [0.567]
CXT [0.558]
CSK [0.551]
TLD [0.549]


Precision plots of TRE – out of view (6)
Precision
Location error threshold
StruckLS [0.488]
TLD [0.486]
CPF [0.473]
LOT [0.463]
Struck [0.459]
CXT [0.444]
VTS [0.431]
VTD [0.420]
TM–V [0.411]
Frag [0.386]





Considering all these plots, we can see that in the general case StruckLS outperforms the state-of-the-art. For us, it is an empirical validation of the presented concept.

Part III
Conclusion

Conclusion

To conclude let us review the proposed evaluations and contributions. We started this project with the goal of studying the feature description in the tracking and try to improve a state-of-the-art technique.

We first try usual keypoint descriptors to describe a patch. But due to the computational cost, we decide to abandon the keypoint descriptor. We do not say that keypoints are not suitable for the tracking, but that they are not so good for patch description.

To keep the advantage of binary description, and address the problem of the number of features, we propose a method to select an efficient subset. The results validate this approach. We absolutely do not say that the selection is optimal, but there is already a significant improvement. With this technique the number of features can be chosen in function of the computational power available, which is a great advantage.

To validate the technique, we integrated it in a complete algorithm, but the technique can be integrated in any tracker by detection.

The other improvements (scale invariance, context awareness) are very specific to the problem of visual tracking, but the feature selection as proposed by Paratte [55] and which is applied in this work is very promising and could be applied to other problems.

The last thing we want to highlight, is that it is really difficult to compare and evaluate the different tracking techniques. The work of Wu and al. [74] is a really good start, but we really need a bigger dataset, with a standard framework for the evaluation (one set of parameters for all the video and a standard scoring technique). Now there are too many publications that test their algorithm on five or ten videos, with very tuned parameters.

Future work

Many aspects of this work can lead to extensions, but here are some interesting aspects that would benefit of more research and we want to explore certain areas soon.

Feature selection

Selection technique

This work present one way to select the features, which already provide good results, but there are several other techniques (based on other hypotheses) which deserve to be studied.

Different feature

We explain why binary features work well, but most of desktop computer we have today come with very impressive optimizations to deal with floating point operations, so it would be interesting to apply the same method to the Haar-like feature.

The application domain

It would be interesting to use feature selection on a patch for other application. With the supervised technique it is relatively easy to specialize the feature selection to a particular task.

Rotation invariance

We propose a technique to add some robustness to the rotation, it would be interesting to test if it really works.

ARM architecture

Due to the massive use of binary feature it would be interesting to evaluate the performance on an ARM architecture.

Technical remarks

The different methods proposed in this work have been implemented using Open Source libraries and Software, and the code can be made available upon request, all the code based on Struck is available on GPLv3, all my contribution is in GPLv3 too. The implementation was done in C++. The OpenCV library[6] was used for almost all vision part and Eigen[7] for the linear algebra.

[5] http://www.gnu.org/licenses/gpl.html
[6] http://opencv.org/
[7] http://eigen.tuxfamily.org

Acknowledgements

Working on this project has been very exciting all along. It is very enjoyable to work on real world problematic. There are some many thing to do in visual tracking.

I would first like to thank Prof. Pierre Vandergheynst and Johan Paratte for making this project possible and for all the help during all the semester.

I would like to thank Sam Hare for resleasing the code of Struck [27] in GPL v3, which opens research and fosters innovation.

I would also like to acknowledge Matthias Brändli and Johan Paratte for their very helpful LaTeXtemplate.

To finish I would like to thank Giorgio Margaritondo for giving me a second chance.

[7] `http://www.gnu.org/licenses/gpl.html`

Equipment and software

Hardware

In this project the following development platforms were used :

Laptop Computer Asus N76VZ

- CPU : Intel Core i7 3630QMK, 4 cores (8 threads) @ 2.4 GHz
- Memory : 16GB RAM @ 1600 MHz
- GPU : nVidia GeForce GT 650M mit 4GB
- Base system : Archlinux 64bit

Desktop Computer Xeon E5-2687W

- CPU : Intel Xeon E5-2687W v2, 16 cores (32 threads) @ 4.5 GHz
- Memory : 128GB RAM @ 1600 MHz
- GPU : Asus GeForce GTX 780
- Base system : Ubuntu 13.10 64bit
- Virtual machine : VirtualBox Windows 7 professional

Software

The following software and libraries were used :

- GCC 4.8.2 with glibc 2.18
- QtCreator 3.0.0
- OpenCV library 2.4 and 3.0
- Eigen 3.2

References

[1] Alahi, A., Ortiz, R., and Vandergheynst, P. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 510–517.

[2] Alt, N., Hinterstoisser, S., and Navab, N. Rapid selection of reliable templates for visual tracking. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 1355–1362.

[3] Avidan, S. Support vector tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 8 (2004), 1064–1072.

[4] Avidan, S. Ensemble tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 29*, 2 (2007), 261–271.

[5] Babenko, B., Yang, M.-H., and Belongie, S. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 983–990.

[6] Babenko, B., Yang, M.-H., and Belongie, S. Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 33*, 8 (2011), 1619–1632.

[7] Bai, Y., and Tang, M. Robust tracking via weakly supervised ranking svm. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 1854–1861.

[8] Bao, C., Wu, Y., Ling, H., and Ji, H. Real time robust l1 tracker using accelerated proximal gradient approach. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 1830–1837.

[9] Bay, H., Tuytelaars, T., and Van Gool, L. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[10] Black, M. J., and Jepson, A. D. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision 26*, 1 (1998), 63–84.

[11] Blaschko, M. B., and Lampert, C. H. Learning to localize objects with structured output regression. In *Computer Vision–ECCV 2008*. Springer, 2008, pp. 2–15.

[12] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010*. Springer, 2010, pp. 778–792.

[13] Collins, R. T., Lipton, A. J., Fujiyoshi, H., and Kanade, T. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE 89*, 10 (2001), 1456–1477.

[14] Collins, R. T., Liu, Y., and Leordeanu, M. Online selection of discriminative tracking features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 27*, 10 (2005), 1631–1643.

[15] Comaniciu, D., Ramesh, V., and Meer, P. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* (2000), vol. 2, IEEE, pp. 142–149.

[16] Comaniciu, D., Ramesh, V., and Meer, P. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 25*, 5 (2003), 564–577.

[17] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[18] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (2005), vol. 1, IEEE, pp. 886–893.

[19] Dinh, T. B., Vo, N., and Medioni, G. Context tracker: Exploring supporters and distracters in unconstrained environments. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 1177–1184.

[20] Fan, J., Wu, Y., and Dai, S. Discriminative spatial attention for robust tracking. In *Computer Vision–ECCV 2010*. Springer, 2010, pp. 480–493.

[21] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 32*, 9 (2010), 1627–1645.

[22] Ferre, M. R., Jakab, P. D., and Tieman, J. S. Position tracking and imaging system with error detection for use in medical applications, Oct. 14 1997. US Patent 5,676,673.

[23] Grabner, H., Grabner, M., and Bischof, H. Real-time tracking via on-line boosting. In *BMVC* (2006), vol. 1, p. 6.

[24] Grabner, H., Leistner, C., and Bischof, H. Semi-supervised on-line boosting for robust tracking. In *Computer Vision–ECCV 2008*. Springer, 2008, pp. 234–247.

[25] Grabner, H., Matas, J., Van Gool, L., and Cattin, P. Tracking the invisible: Learning where the object might be. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 1285–1292.

[26] Hager, G. D., and Belhumeur, P. N. Efficient region tracking with parametric models of geometry and illumination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 20*, 10 (1998), 1025–1039.

[27] Hare, S., Saffari, A., and Torr, P. H. Struck: Structured output tracking with kernels. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 263–270.

[28] Haritaoglu, I., Harwood, D., and Davis, L. S. W< sup> 4</sup>: real-time surveillance of people and their activities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 22*, 8 (2000), 809–830.

[29] Hearst, M. A., Dumais, S., Osman, E., Platt, J., and Scholkopf, B. Support vector machines. *Intelligent Systems and their Applications, IEEE 13*, 4 (1998), 18–28.

[30] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. Exploiting the circulant structure of tracking-by-detection with kernels. In *Computer Vision–ECCV 2012*. Springer, 2012, pp. 702–715.

[31] Hu, W., Tan, T., Wang, L., and Maybank, S. A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 34*, 3 (2004), 334–352.

[32] Hu, Y., Zhao, W., and Wang, L. Vision-based target tracking and collision avoidance for two autonomous robotic fish. *Industrial Electronics, IEEE Transactions on 56*, 5 (2009), 1401–1410.

[33] Iannizzotto, G., and La Rosa, F. Competitive combination of multiple eye detection and tracking techniques. *Industrial Electronics, IEEE Transactions on 58*, 8 (2011), 3151–3159.

[34] Isard, M., and Blake, A. Condensation—conditional density propagation for visual tracking. *International journal of computer vision 29*, 1 (1998), 5–28.

[35] Jepson, A. D., Fleet, D. J., and El-Maraghi, T. F. Robust online appearance models for visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 25*, 10 (2003), 1296–1311.

[36] Jia, X., Lu, H., and Yang, M.-H. Visual tracking via adaptive structural local sparse appearance model. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 1822–1829.

[37] Kalal, Z., Matas, J., and Mikolajczyk, K. Pn learning: Bootstrapping binary classifiers by structural constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 49–56.

[38] Kass, M., Witkin, A., and Terzopoulos, D. Snakes: Active contour models. *International journal of computer vision 1*, 4 (1988), 321–331.

[39] Kwon, J., and Lee, K. M. Visual tracking decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 1269–1276.

[40] Kwon, J., and Lee, K. M. Tracking by sampling trackers. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 1195–1202.

[41] Learned-Miller, E. G., and Lara, L. S. Distribution fields for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition* (2012).

[42] Leistner, C., Saffari, A., Roth, P. M., and Bischof, H. On robustness of online boosting-a competitive study. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 1362–1369.

[43] Leutenegger, S., Chli, M., and Siegwart, R. Y. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 2548–2555.

[44] Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., and Hengel, A. v. d. A survey of appearance models in visual object tracking. *arXiv preprint arXiv:1303.4803* (2013).

[45] Li, Y., Ai, H., Yamashita, T., Lao, S., and Kawade, M. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 30*, 10 (2008), 1728–1740.

[46] Lienhart, R., and Maydt, J. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on* (2002), vol. 1, IEEE, pp. I–900.

[47] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (Nov. 2004), 91–110.

[48] Lucas, B. D., Kanade, T., et al. An iterative image registration technique with an application to stereo vision. In *IJCAI* (1981), vol. 81, pp. 674–679.

[49] Masnadi-Shirazi, H., Mahadevan, V., and Vasconcelos, N. On the design of robust classifiers for computer vision. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 779–786.

[50] Matthews, L., Ishikawa, T., and Baker, S. The template update problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 6 (2004), 810–815.

[51] Mei, X., and Ling, H. Robust visual tracking using l1 minimization. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 1436–1443.

[52] Mei, X., Ling, H., Wu, Y., Blasch, E., and Bai, L. Minimum error bounded efficient l1 tracker with occlusion detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 1257–1264.

[53] Mei, X., Ling, H., Wu, Y., Blasch, E., and Bai, L. Efficient minimum error bounded particle resampling l1 tracker with occlusion detection.

[54] Ojala, T., Pietikäinen, M., and Harwood, D. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition 29*, 1 (1996), 51–59.

[55] Paratte, J. Sparse binary features for image classification.

[56] Pérez, P., Hue, C., Vermaak, J., and Gangnet, M. Color-based probabilistic tracking. In *Computer vision—ECCV 2002*. Springer, 2002, pp. 661–675.

[57] Porikli, F., Tuzel, O., and Meer, P. Covariance tracking using model update based on lie algebra. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (2006), vol. 1, IEEE, pp. 728–735.

[58] Ross, D. A., Lim, J., Lin, R.-S., and Yang, M.-H. Incremental learning for robust visual tracking. *International Journal of Computer Vision 77*, 1-3 (2008), 125–141.

[59] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 2564–2571.

[60] Saffari, A., Godec, M., Pock, T., Leistner, C., and Bischof, H. Online multi-class lpboost. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 3570–3577.

[61] Saffari, A., Leistner, C., Godec, M., and Bischof, H. Robust multi-view boosting with priors. In *Computer Vision–ECCV 2010*. Springer, 2010, pp. 776–789.

[62] Santner, J., Leistner, C., Saffari, A., Pock, T., and Bischof, H. Prost: Parallel robust online simple tracking. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 723–730.

[63] Stenger, B., Thayananthan, A., Torr, P. H., and Cipolla, R. Model-based hand tracking using a hierarchical bayesian filter. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 28*, 9 (2006), 1372–1384.

[64] Stenger, B., Woodley, T., and Cipolla, R. Learning to track with multiple observers. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 2647–2654.

[65] Tang, F., Brennan, S., Zhao, Q., and Tao, H. Co-tracking using semi-supervised support vector machines. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (2007), IEEE, pp. 1–8.

[66] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research* (2005), pp. 1453–1484.

[67] Tuzel, O., Porikli, F., and Meer, P. Region covariance: A fast descriptor for detection and classification. In *Computer Vision–ECCV 2006*. Springer, 2006, pp. 589–600.

[68] Vedaldi, A., Gulshan, V., Varma, M., and Zisserman, A. Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 606–613.

[69] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (2001), vol. 1, IEEE, pp. I–511.

[70] Viola, P., and Jones, M. J. Robust real-time face detection. *International journal of computer vision 57*, 2 (2004), 137–154.

[71] Wang, D., Lu, H.-C., and Yang, M.-H. Online object tracking with sparse prototypes.

[72] Wang, Y., Teoh, E. K., and Shen, D. Lane detection and tracking using b-snake. *Image and Vision computing 22*, 4 (2004), 269–280.

[73] Wu, Y., Cheng, J., Wang, J., Lu, H., Wang, J., Ling, H., Blasch, E., and Bai, L. Real-time probabilistic covariance tracking with efficient model update. *Image Processing, IEEE Transactions on 21*, 5 (2012), 2824–2837.

[74] Wu, Y., Lim, J., and Yang, M.-H. Online object tracking: A benchmark.

[75] Wu, Y., Lin, J., and Huang, T. S. Analyzing and capturing articulated hand motion in image sequences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 27*, 12 (2005), 1910–1922.

[76] Wu, Y., Ling, H., Yu, J., Li, F., Mei, X., and Cheng, E. Blurred target tracking by blur-driven tracker. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 1100–1107.

[77] Yang, M., Wu, Y., and Hua, G. Context-aware visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 31*, 7 (2009), 1195–1209.

[78] Yang, S. X., Zhu, A., Yuan, G., and Meng, M. A bioinspired neurodynamics-based approach to tracking control of mobile robots. *Industrial Electronics, IEEE Transactions on 59*, 8 (2012), 3211–3220.

[79] Yoon, J. H., Yoon, K.-J., et al. Visual tracking via adaptive tracker selection with multiple features. In *Computer Vision–ECCV 2012*. Springer, 2012, pp. 28–41.

[80] Zeisl, B., Leistner, C., Saffari, A., and Bischof, H. On-line semi-supervised multiple-instance boosting. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 1879–1879.

[81] Zhang, L., Chu, R., Xiang, S., Liao, S., and Li, S. Z. Face detection based on multi-block lbp representation. In *Advances in Biometrics*. Springer, 2007, pp. 11–18.

[82] Zhang, T., Ghanem, B., Liu, S., and Ahuja, N. Robust visual tracking via multi-task sparse learning. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 2042–2049.

[83] Zhang, X., Hu, W., Chen, S., and Maybank, S. Graph embedding based learning for robust object tracking.