Robust and Long-Term Object Tracking With an Application to Vehicles

Feng Zheng, Ling Shao, *Senior Member, IEEE*, and Junwei Han

***Abstract*— Recently, intelligent vehicles catch much attention in both academia and industry. The vision-based moving object/vehicle detection and tracking are typically the core techniques for the event and activity analysis and the understanding of the dynamic driving environment in an intelligent vehicle. However, due to the complicated non-stationary environment, most existing vision-based motion tracking algorithms proposed for other simple conditions are not able to consistently track the objects. Therefore, in this paper, we propose a robust and long-term tracking method for intelligent vehicles, in which a set of classifiers are dynamically maintained and sampled for tackling varied challenges. In contrast to previous methods, to increase the diversity, a set of basic classifiers trained sequentially on different small data sets over time is dynamically maintained. The subsets of basic classifiers are independent with each other and can be specified to solve certain different sub-problems occurred in a non-stationary environment. Thus, for every challenge, an optimal classifier can be approximated in a subspace spanned by the selected competitive classifiers, which can address the current problem according to the distribution of the samples and recent performance. As a result, the tracker can efficiently address the various "concept drift" problems occurred together in a long video sequence. Due to the use of sparse weights for the competitive classifiers, the tracker can keep the balance between the efficiency and the performance. Experimental results show that the tracker yields competitive performance under various challenging environmental conditions and, especially, can overcome several challenges simultaneously.**

***Index Terms*— Motion tracking, concept drift, on-line learning, intelligent vehicle.**

I. Introduction

INTELLIGENT vehicles are complex and integrated systems consisting of various sensors and computational modules to automatically reduce risks [1], [2], high accidents rate and traffic congestion, when the vehicles are moving on roads. Undoubtedly, among them, vision-based motion tracking is a vital technique, because of the capability of computer vision enhanced cameras and the low-cost advantage [3]. Actually, visual target tracking is a traditional and fundamental topic and has wide-ranging applications in designing various systems such as surveillance, augmented reality, robotics and human-computer interaction. However, to date, developing an effective and efficient vision-based target tracking method for intelligent vehicles is still challenging, due to the non-stationary environment (NSE) surrounding vehicles, such as presence of occlusion, background clutter, varying viewpoints, illumination changes, scale changes and camera motion.

Recently, adaptive tracking-by-detection approaches [4]–[9] based on machine learning methods, which treat the tracking problem as a classification task, are proposed to overcome difficulties in the NSE. Most machine learning algorithms can learn from data that are assumed to be drawn independently from a fixed but unknown distribution (i.i.d.). However, the i.i.d. assumption cannot be valid in case of the tracking problem, particularly in the view of moving vehicles. This condition results in that the classifiers for tracking can only be trained on a small sample set and, more importantly, these samples captured in the view of moving vehicles are generated over time. Thus, traditional machine learning methods applied to the tracking problem will fail when there is a "concept drift" in the NSE. The term "***concept drift***" can be used to represent ***changes*** in the underlying ***distribution of samples***. In motion tracking, the distribution of samples changes a lot due to the deformation of the object and the change in the underlying environment. Especially during the transition between different difficulties (sub-problems), such as from occlusion to varying viewpoints, the samples in the two different situations differ significantly. Thus, the function learned on a fixed sample set previously collected may not reflect the current state of nature and the separability of adopted features will decrease in the new situation. If x are the samples and $y \in \{1, -1\}$ are classes, the whole distribution of the problem at time t, which is characterised by the joint distribution $p^t(x, y)$, can be represented by: the unconditional probability density function $p^t(x)$ and posterior probabilities $p^t(y|x)$. Then, the "concept drift" can be defined as any scenarios where the posterior probability changes over time:

$$KL(p^{t+n}(y|x), p^t(y|x)) < \tau_d \tag{1}$$

where t is the time, n is the time step of drift, τ_d is a small value and the Kullback-Leibler (KL) divergence describes the dissimilarity of the two distributions.

Manuscript received March 27, 2017; revised May 29, 2017; accepted July 10, 2017. The Associate Editor for this paper was Q. Wang. *(Corresponding author: Ling Shao.)*

F. Zheng is with the School of Automation, Northwestern Polytechnical University, Xi'an 710065, China, and also with the Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield S10 2TN, U.K.

L. Shao is with the School of Automation, Northwestern Polytechnical University, Xi'an 710065, China, and also with the School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, U.K. (e-mail: ling.shao@ieee.org).

J. Han is with the School of Automation, Northwestern Polytechnical University, Xi'an 710065, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2017.2749981

Fig. 1. A sample view of moving vehicle to capture a motor. Assuming that the best classifiers for the previous frames are available, which classifiers should be used in the current frame (bottom right)? f_2, f_5 or their combination? Also, when the target moves out of view then comes back, which classifiers are the best to be used? This paper tries to solve these problems in motion tracking.

In this paper, we first empirically investigate the "concept drift" problems in motion tracking. Then, according to the findings of analysis, a Learn++ (LPP) tracker is proposed to dynamically sample competitive classifiers for robust and long-term motion tracking. To increase the efficiency and stability for the model, a competitive strategy is adopted to separately solve various "concept drift" challenges appeared in a same video sequence. Learn++ [10] is a new group of machine learning methods to learn additional information from new data without accessing the original samples and can be used for recognition tasks in very complex situations where new classes would join in. The structure of an optimal classifier in Learn++ is very similar to that in AdaBoost but there are several key differences: AdaBoost runs on a single database and is based on the assumption of an i.i.d. distribution; Some classifiers whose errors are larger than 0.5 will be discarded; Most importantly, the later weak classifier in AdaBoost depends on their previous one; And, the weight distribution of samples is updated using the error of current weak classifier. However, Learn++ can address a set of databases in which the samples are generated by different distributions. To increase the diversity of the model, Learn++ keeps all the classifiers as long as they can achieve good performance on a subset. In learn++, the weights of samples are updated using the ensemble performance.

The proposed LPP tracker dynamically maintains a set of basic classifiers $\Omega_e^t = \{f_i\}$ which are trained sequentially on a small sample set. The "concept drift" problems can be solved by adaptively sampling the most suitable classifiers named as competitive subset $\Omega_a^t \subset \Omega_e^t$ as shown in Fig. 1. These basic classifiers are independent from each other and used to address different sub-problems. For each challenge, the democratic mechanism can be adopted, where all classifiers should compete with each other to be added into a competitive subset to suit the present environment. Next, the optimal classifier $\mathbf{f}^t$ in the present environment can be fast searched in a function space linearly spanned by these basic classifiers in the competitive subset. After the detection guided by motion constraints, the most important samples will be collected to update the classifiers which are trained in the same situation.

In summary, we make the following three contributions: (1) Empirical analysis, which concludes that motion tracking is a non-i.i.d. sampling and small dataset problem, is given to guide the design of a motion tracker for the vehicle intelligent system. (2) A new framework of Learn++ particularly for motion tracking is proposed. Unlike classical Learn++ methods, a competitive subset of classifiers which consists of the ones more adaptive to the current environment is maintained, a constraint (motion) is added to guide the learning and new samples are only used to update these classifiers trained in similar situations. (3) As far as we know, LPP tracker is the first tracking method that designs an explicit model for each sub-problem (i.e., challenge) and the models can be automatically altered according to the environment. The most distinctive merit of LPP tracker is that, even when a target moves out of view and then comes back with totally different locations and appearances, the tracker can still lock the target as long as the appearances ever appeared in the past. Because of training a relatively large set of classifiers, LPP tracker keeps its diversity so as to improve the generalization and performance no matter what concept drifts occur. Despite the complexity of the model, due to the sparsity of competitive set, LPP tracker is still a real-time method.

The rest of the paper is organised as follows. We first review the related work in Section II. A fast and compact descriptor for image patches is introduced in Section III. The empirical analysis is given Section IV. Section V details the proposed Learn++ based method for visual tracking. Experimental results are reported and analysed in Section VI. Finally, conclusions are drawn in Section VII.

II. Related Work

Generally, according to the type of samples used to build the model, the online adaptive algorithms can be divided into two groups: generative methods which mostly focus on positive samples to infer the relationship between them, and discriminative methods which use both positive and negative samples to train a classification hyperplane.

A most basic concept of motion tracking is direct image patch matching. Following this basic idea, there are several well-known methods, such as: Lucas-Kanade tracker [11], fragments-based tracker [12] and mean shift tracking [13]. However, the target in these methods is not updated according to the appearance change of the object. Particularly the saliency [14] of the target is not consistent during tracking. Thus, an essential step forward is to build a generative appearance model to capture the variation over time, such as online subspace learning [5] and sequential Monte Carlo sampling [15]. Recently, sparse coding based methods catch much attention in the community of object tracking. Reported in the two experimental evaluation results [16], [17], the sparse coding based methods [18], [19] achieve high performance. However, despite the superior performance on partial occlusion, in a survey [20], the authors state that their experimental results have shown that visual tracking may not be a sparse representation problem. Moreover, generative methods would easily fail with a cluttered background.

Considering the significant role of discriminative information from background, pioneered by support vector tracker [21] and ensemble tracker [7], various discriminative algorithms have been built to model the difference between the foreground and the background. Collins *et al.* [6] explore a mechanism which adaptively selects the most discriminative features from a set of different colour spaces. In addition, the random projection is used in compressive tracking (CT) [22]. However, CT is a data-independent method which guarantees that no noise is introduced but lacks flexibility. Moreover, numerous methods exploring the different properties of samples and relationships between samples, including P-N learning [23], semi-supervised SVM [24], semi-supervised boosting [25], multiple instance learning [4], weighted reservoir sampling [26] and semi-supervised transfer learning [27], have been also proposed to improve the performance of trackers. Recently, in [28], the confidence of a classifier is considered as a probability which can be analysed using Gaussian Processes regression. Structured output tracking with kernels (Struck) [29], using the windows as input, explores the training data with the form of appearance and translation. The experimental survey [30] concludes that Struck performs well on all aspects but one, the change of scale, bringing it to the number one position over their entire dataset.

Very recently, several methods adopt the concept of multiple-expert system to improve the performance. It is worth to mention that, although AdBoost consists of a number of weak classifiers, the AdBoost based trackers [7], [31] are not multiple-expert systems because they are still realised using one strong classifier. In PROST [8], the Lucas-Kanade (LK) [11] method and one random forest based classifier are combined for target tracking. A major difference between PROST and our LPP tracker is that our proposed method dynamically samples a set of classifiers to increase the diversity of the model. Similar to PROST, Yan *et al.* [32] design an ensemble framework for the optimal selection of detectors and trackers to do multi-target tracking. To further improve the robustness of tracker, the depth camera can be used. Some features including optical flow, color and depth clues are simultaneously considered in [33] and [34] fuses the depth-map and the mid-level features captured by superpixels. Visual tracker sampler (VTS) [35] incorporates a process of sampling trackers into the framework of particle filtering [36], without differentiating the trackers. In contrast, each member of LPP tracker, which is trained on different sample sets and learned online under constraints, is inclined to be more independent. Yoon *et al.* [37] adopt two steps: tracker selection and interaction to the multiple features fusion. LPP tracker maintains a set of classifiers corresponding to different features. The feature fusion and selection are processed by sampling the classifiers. Randomised ensemble trackers (RET) [38] consider the weights of classifiers as a non-stationary distribution. The difference is that RET approximates weights of classifiers by sampling the Dirichlet distribution while LPP tracker approximates weights based on recent performance and the manifold structure of training samples.

All these methods require various updating schemes to capture the continuous deformations of the objects.

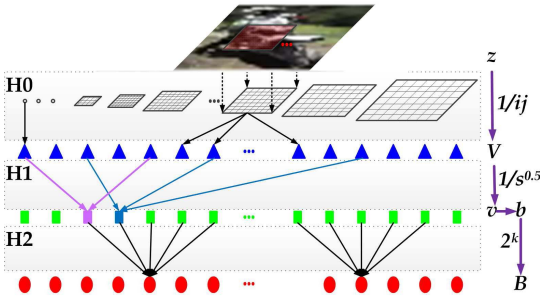


Fig. 2. The framework of fast structural representation can be chosen by our proposed LPP tracker.

As a consequence, they tend to drift by incorporating wrong information [9]. To avoid the drifting, diverse strategies are adopted, such as: different update rates [39], data-independent knowledge [22] and selectively updating the parts [40]. However, the essential reason why drifting occurs is that classical trackers have not considered the object tracking as a "concept drift" problem and tried to solve different challenges by only one super-power model. In fact, the differences between various challenges are very large. Thus, we can see the limitation of the classical trackers comes from the basic i.i.d. assumption in machine learning, on which most of tracking-by-detection methods depend. The drifting problem is not very obvious and can be partially solved by the classical methods in short sequences but it is still quite difficult for the long-term tracking [30], [41]. In the tracking-by-detection methods, recovering from drift may also prove a useful way to make tracking robust but the update of wrong information will destroy the structure of the classifier.

III. Structural Representation for Image Patch

In motion tracking, the two important factors of appearance representation are efficiency and discriminativeness. At present, lots of powerful features have been explored for various tasks but most of them cannot be directly used in motion tracking, because they are either computationally expensive or not discriminative in NSE. In this section, a fast structural representation (SR) is introduced to represent an image patch as shown in Fig. 2. The advantage of scale invariant structural representation is that the optimal projections and filters can be decided by selecting the third-layer nodes using the specially designed classifier. As a result, the proposed tracker can keep a good balance between efficiency and performance. Structural representation has a three-level hierarchy: H0-virtual stage of filtering, H1-stage of random projection, and H2-stage of encoding.

A. H0: Filtering

Given a patch $z \in R^{I \times J}$, a set of rectangular smoothing filters $\{h_{i\times j} \in R^{i\times j}, 1 \le i \le I, 1 \le j \le J\}$ are defined, for which all entries of each filter $h_{i\times j}$ equal $1/(i \times j)$. In total, there are $I \times J$ filters, each of which is convolved with the entire patch and produces $I \times J$ values. So, the dimension $n_V = (IJ)^2$ of the original feature $V \in R^{n_V}$ is very high and much information is redundant.

B. H1: Random Projection

Next, a sparse random matrix is used for dimensionality reduction, which is defined as: $P(i, j) = 1$ with the probability $1/2s$, $P(i, j) = -1$ with the probability $1/2s$ and $P(i, j) = 0$ with the probability $1 - 1/s$, where p is the probability. In [42], Achlioptas pointed out that this matrix with $s = 2$ or 3 satisfies the Johnson-Lindenstauss lemma. Compressive sensing theory ensures that the extracted features preserve almost all the information of the original image patch. In this paper, we set $s = n_V/4$.

Thus, the value $v_k \in R$ projected by each row of the random matrix is: $v_k = P(k, \cdot)V$. The stage of filtering can be considered as virtual. This is because most of the entries are zeros so that a large proportion of the filter needs not to be calculated. We only need to store the nonzero entries and the positions of their corresponding rectangular filters in an image. Moreover, v_k can be efficiently computed by using $P(k, \cdot)$ to sparsely measure the rectangular features which can be efficiently computed using the Integral Image. For patches with a different size $z^* \in R^{I^* \times J^*}$, the number of rectangular features will be different. In fact, we need not to resize the patch. Applying a scale $IJ/(I^*J^*)$ to the locations of elements in V^* will be feasible to realize scale invariance. For each value v_k, its mean μ_k and variance σ_k of positive samples will be computed when its corresponding classifier is trained.

C. H2: Encoding

The third layer is constructed similarly to Fern [43], in which a feature was calculated by comparing two randomly selected pixels in a patch. However, directly comparing two pixels is very sensitive to noise, especially when the two pixels are located around an edge. Normally, to eliminate this drawback, filters will be used firstly. Instead of comparing the pixels, the value v_k can be considered as basic cues. Thus, for each projected value v_k, a binary feature can be defined as: $b_k = \Gamma\lfloor v_k \in [\mu_k - \sigma_k, \mu_k + \sigma_k]\rfloor$, where $\Gamma\lfloor\rfloor$ is the indicative function. Each node in the third layer consists of a set of bits, and in this paper, the size of the set n_b is set to 7. Thus, the node of the deepest layer in the hierarchy is defined as: $B = \sum_{k=0}^{n_b} 2^k b_k$.

In summary, firstly, structural representation is a simple but powerful and efficient representation for image patch, which has a similar formulation to the convolutional neural network algorithm [44] in the first several layers. We can see that the first layer is to calculate local mean values with various scales and the second layer, which combines patches in different locations, is to capture the global structure of objects. Thus, structural representation can extract the local and global features simultaneously and be invariant to partial occlusion. The binary encoding in the third layer, to which a naive Bayes classifier can be directly used, plays the same role as activation function. Secondly, if the values of sparse projection are fixed so that adjacent rectangles are combined and no binary coding is used, then structural representation extracts the Harr-like features. Next, if the size of a filter is fixed, the number of nonzero entries of random projection is set to two, and the two weights are also opposite numbers, then structural representation will become the classical framework of Fern [43]. Moreover, besides the natural properties of intensity, scale and partial occlusion invariance, structural representation is very computationally efficient because of the usage of sparse projection and binary coding. Finally, each node in the third layer can capture a certain internal structure of objects. By selecting the nodes, different information can be used. Therefore, various challenges in object tracking can be handled by selecting the abundant and diversified features.

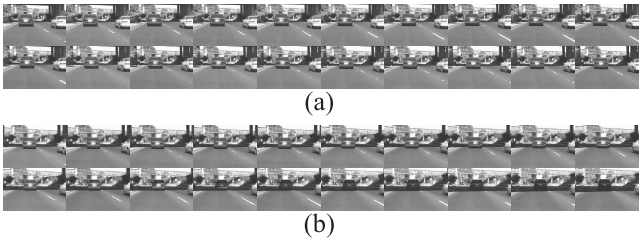


Fig. 3. Two short videos in Car4 [45] to show the difference between the trivial case (a) and drift case (b). Each short video consisting of **front set** (first row) and **latter set** (second row) is annotated by three numbers. Take the second video for example, the first number #173 is the index of the starting frame, the second one #182 denotes the moment when the object begins to change and the third one #192 is the index of the ending frame. (a) Trivial case with frames from #143 to #162. (b) Drift case with frames from #173 to #192.

IV. "Concept Drift" in Motion Tracking

In this section, the "Concept Drift" problem in motion tracking is investigated on a public dataset which contains 50 sequences [45]. From the theoretical definition of "Concept Drift", to investigate the "drift", the necessary steps are firstly to fit the **sample distribution** and then to calculate the **changes** of the distribution over time. However, it is impractical to directly obtain stable distributions, because, normally, the dimension of a sample is very high. Therefore, we seek to calculate the distribution of Euclidean distance between any pair of samples. The intuition behind is that the distribution of distances or similarities can reflect the distribution of samples. For example, if two variables x_1 and x_2 are independent, standard normal random variables, then the quantity $||x_1 - x_2||^2$ is distributed according to the one degree chi-squared distribution.

Moreover, we observe that most classical methods can handle the translation in simple situations (trivial cases shown in Fig. 3 (a)) without large deformation or other challenges, but are unable to cope with complex "concept drift" situations (drift cases shown in Fig. 3 (b)). However, at present, the differences of appearance features between the trivial and the drift situations have not been studied. The study of the differences can facilitate understanding the nature of motion tracking and guide the design of more robust methods. To achieve this, 100 short videos (A sample is shown in Fig. 3 (b)) with the challenges excluding fast motion[1] from the dataset are manually identified and other 984 short videos (A sample is shown Fig. 3 (a)) are randomly selected from

[1]This problem can be solved easily by dense sampling using the tracking-by-detection methods.

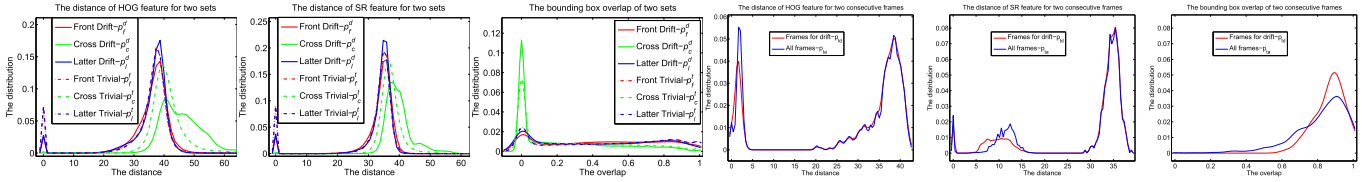


Fig. 4. Statistical analysis of the "concept drift" situation and the trivial situations in object tracking. (a) The distance distribution of the HOG feature between the two sets. (b) The distance distribution of the structural representation feature between the two sets. (c) The bounding box overlap distribution between the two sets. (d) The distance distribution of the HOG feature for two consecutive frames. (e) The distance distribution of the structural representation feature for two consecutive frames. (f) The bounding box overlap distribution of two consecutive frames.

TABLE I

THE DISCREPANCY $\int_d |p_1(d) - p_2(d)| \bigtriangledown d$ BETWEEN THE TWO DISTRIBUTIONS $p_1(d)$ AND $p_2(d)$ SHOWN IN FIG. 4. THE LARGEST DISCREPANCY APPROXIMATES 2

Comparison	p_f^d p_f^t	p_c^d p_c^t	p_l^d p_l^t	p_t^d p_{ta}	p_c^d p_{td}
HOG feature	0.2326	0.7488	0.1835	0.1408	1.2590
SR feature	0.2142	0.5976	0.1548	0.1817	1.1441
Bounding box	0.1579	0.2037	0.1423	0.3518	0.6858

the rest of the dataset. Thus, we can see that the short videos are divided into two parts by the three numbers: the front set before drift and the latter set after the drift.

Firstly, based on the two sets, three groups of feature distances $d = ||x_1 - x_2||$ can be calculated depending on which sets the pairs of samples come from. Then, for each group of distances, a distribution of them can be obtained. For the "drift" cases, p_f^d denotes the distribution of distances between two samples both within the front set, p_l^d denotes the distribution of distances between two samples both within the latter set and p_c^d denotes the distribution of distances across the two sets. For the trivial cases, we can also calculate the three distance distributions including p_f^t, p_l^t and p_c^t. In this analysis, two features, HOG [46] and structural representation, are used. The comparisons of six distributions are shown in Fig. 4 (a) and (b) for HOG and structural representation, respectively. Moreover, besides the feature distance, the distribution of the overlap of bounding boxes in the two sets is also investigated. The comparisons of six distributions for the overlap of bounding boxes are given in 4 (c). In contrast, for the feature distance and overlap of bounding boxes between any two consecutive frames, distributions of them are also calculated and the comparisons are referred to 4 (d), (e) and (f), respectively. Finally, the quantitative analysis $\int_d |p_1(d) - p_2(d)| \bigtriangledown d$ between two distributions are given in Table I.

From the statistical analysis given in Fig. 4 and Table I, the following observations can be obtained:

- The six distributions of the two types of features HOG and structural representation are similar. It indicates that the statistics reflect the nature of variations of objects and the environment.
- The feature distances across the two sets are obviously larger than the ones within a set. Next, from Table I, we can see the distribution discrepancy between p_c^d and p_c^t both across the two sets is almost three times larger than the one between p_f^d and p_f^t within the front set. Furthermore, the distribution discrepancy between p_f^d within set before drift and p_c^d across the two sets is quite large. Because the distance distribution can reflect the intrinsic structure of samples, the analysis demonstrates that the sample distributions of the two sets distinguish from each other, especially for the cases of "concept drift" situations. To some extent, the assumption that the samples are generated by different distributions over time is verified.
- In contrast, the distributions of feature distance between the two consecutive frames for the drift and trivial conditions are similar and the distribution discrepancy is also small, no matter which type of representation is adopted.
- Furthermore, the overlap distributions of bounding boxes between the two consecutive frames are also similar for the drift and trivial situations.

In brief, by the empirical study, we can conclude that the motion tracking is a non-i.i.d. sampling and small set problem. Therefore, the ability of classical machine learning is limited. And, it is unsuitable to classify the samples in the following set after the "concept drift", by a function which is trained on a small sample set generated by different latent distributions. However, the procedure of tracking-by-detection can be connected together using the motion information (two consecutive frames). In this paper, a competing strategy is adopted to select the most suitable classifiers to address the current special problem and the motion constraint between two consecutive frames is used to supervise the training and updating of classifiers.

V. LEARN++ FOR SOLVING THE PROBLEM OF "CONCEPT DRIFT"

Based on the above observations, in this section, a LPP tracker is learned to solve the numerous problems of "Concept Drift" which is guided by motion constraints. Assume the classifier set Ω_e^t consists of a competitive set Ω_a^t and its complementary set Ω_c^t. We have $\Omega_c^t \bigcup \Omega_a^t = \Omega_e^t$, $n_e^t = |\Omega_e^t|$ and $n_a^t = |\Omega_a^t|$, where $|\cdot|$ denotes the number of members of the set. W_i^t denotes the historical weights of all existing $f_i \in \Omega_e^t$.

A. Motion Constraints

The similarity transform is used as the motion model for our system. There are four parameters: (1) horizontal and

vertical coordinates; (2) horizontal and vertical scales. a is the state of target describing its motion.

Based on the observation in two consecutive frames, Optical flow (OP) [11] is used to construct the motion model in our framework. OP follows the movement of the target frame by frame and has very high flexibility because no historical information is considered. The motion model $p(a^t|a^{t-1})$ reflects the motion characteristic of the target by predicting the current state a^t based on the previous state a^{t-1}. If $p(a^t|a^{t-1}) > \tau_1$, then a^t is a valid result no matter whether "concept drift" occurs or not. The P-N constraints [23], $p(a^t|f_i)$, were proposed to estimate the confidence of the ith classifier f_i. If $p(a^t|f_i) > \tau_2(i)$ where $f_i \in \Omega_a^t$, the outcome of classifier f_i is validated. This triggers the application of the P-N constraints that exploit the structure of the data. From the manifold perspective, P-N constraints maintain a purified sub-manifold for positive and negative samples. All samples far from such a sub-manifold will be ignored. So, P-N constraints are used by LPP tracker to guarantee the stability of each classifier. If $p(a^t|\Omega_a^t) > \tau_2$, where $p(a^t|\Omega_a^t) = \max_i p(a^t|f_i)$, there is no occurrence of drifting. The classifier constraints are designed based on the observations of the two sets in the analysis of "concept drift".

B. Objective Function

In frame t, x_l^t and y_l^t denote the structural representation and the label of image patch z_l^t, respectively. Each entry $x_l^t(i)$ contains the n_B number of nodes $\{B_{i,j} : j = 1, \cdots, n_B\}$, for the classifier f_i. Also, X^t is the set of collected samples and $n_X^t = |X^t|$. The distribution of samples D^t will be calculated according to the results of old classifiers and used to describe the importance of samples. For simplicity, we define $f_i(x_l^t) = f_i(x_l^t(i))$, $w^t = (w_1^t, \cdots, w_{n_a}^t)$ and $\Omega_e^t = (f_1, \cdots, f_{n_a})^T$.

Our goal is to find an optimal classifier $\mathbf{f}^t$ with most discriminative features in the function space $\mathcal{H}^t$ linearly spanned by a set of classifiers Ω_e^t which are trained in previous frames, where $\mathcal{H}^t = \{h^t : h^t = w^t \Omega_e^t\}$. Moreover, to improve the efficiency of the system, the weights w^t are required to be sparse so that most basic classifiers are not used in the current frame. Thus, in theory, the objective function is defined as:

$$\mathbf{w}^t = \arg\min_{w^t} \sum_l L(h^t(x_l^t), y_l^t) + \lambda \left|\left|w^t\right|\right|_0, \tag{2}$$

where L and λ are the loss function and regularization parameter, respectively. Therefore, we obtain the hypothesis as:

$$\mathbf{f}^t = \mathbf{w}^t \Omega_e^t \tag{3}$$

The optimal classifier $\mathbf{f}^t$ can be used to detect the object in the current frame (new environment). The final classification for each image patch x_i^t is achieved as: $y_l^t = sign(\mathbf{f}^t(x_l^t))$. Eqn. 2 cannot be optimised directly, due to that the true label y_l^t of image patch x_l^t is unknown. However, based on the assumption of the "concept drift" (Eqn. 1), we can approximate to the optimal solution using the classifiers that have yielded good performance in recent n frames or in the same situations by Learn++. Learn++, which is an ensemble of classifiers originally developed for incremental learning. It specifically seeks the most discriminative information from each sample set (sub-problem) through sequentially generating an ensemble of classifiers which are trained on individual data sources and carry complementary information. It can still achieve a statistically significant improvement by combining these classifiers which are finely tuned for the given problem. In this paper, the "concept drift" problem in the NSE is solved through two steps: (1) selection of the active subset Ω_a^t; (2) optimal approximation $\mathbf{f}^t = \mathbf{w}^t \Omega_a^t$. In the following, how to train basic classifiers f_i and how to approximate the optimal classifier $\mathbf{f}^t$ by calculating w_i^t will be introduced.

C. Basic classifier

The Naïve Bayesian are used as the basic classifiers in our proposed system. Thus, f_i can be defined by posterior probabilities by combining n_B nodes (assuming an uniform prior $p(y)$):

$$f_i(x_l) = \arg\max_y p(y|x_l(i)) \tag{4}$$

where $p(y|x_l(i)) \propto \prod_j^{n_B} p(x_l(i, j)|y)$. Therefore, for each f_i, the posterior probabilities will be trained and updated to adapt to the changes of the environment and the object by calculating and updating the class conditional distribution $p^t(B_{i,j}|y)$ of each Fern.

1) Training: The parameters of structural representation for each classifier f_i will be generated randomly. Once generated, these parameters will be fixed during the whole lifespan of the classifier f_i. At frame t, based on a set X_1^t with all positive samples and 2000 negative samples in the set X^t and distribution D^t, we can define two quantities which are used to train or update classifiers: $N^t(y, B_{i,j}) = \sum_l D_l^t \Gamma \lfloor x_l^t(i, j) = B_{i,j} \rfloor \Gamma \lfloor y_l^t = y \rfloor$ and $N^t(y) = \sum_l D_l^t \Gamma \lfloor y_l^t = y \rfloor$, where $x_l^t \in X_1^t$. Other negative samples are used to evaluate the classifier. Therefore, $\tau_2(i) = max_{a_l^t} p(a^t|f_i)$, where a_l^t denotes the corresponding states of negative samples $x_l^t \in X^t / X_1^t$. Thus, the class conditional distributions for f_i are calculated by:

$$p^t(B_{i,j}|y) = \frac{1 + N(y, B_{i,j})}{1 + N(y)} \tag{5}$$

where $N(y, B_{i,j}) = N^t(y, B_{i,j})$ and $N(y) = N^t(y)$.

2) Learning: If f_i has been used in frame t. The set X^t can be used to update the class conditional distribution $p^t(B_{i,j}|y)$ so as to adapt to the changes by:

$$\begin{aligned} N(y, B_{i,j}) &\Leftarrow N(y, B_{i,j}) + N^t(y, B_{i,j}); \\ N(y) &\Leftarrow N(y) + N^t(y) \end{aligned} \tag{6}$$

By recalculating Eqn. 5, the updated class conditional distributions are obtained.

D. Tracking by Detection

At the beginning ($t = 0$), random Ferns f_1 needs to be trained according to the selected target in the first frame and we can directly jump to the sample collection step. At frame $t\,(t > 0)$, assume that $\mathbf{f}^t(x_l^t)$ is the ensemble learned

at time $t-1$ and the location $\mathbf{a}^{t-1}$ has been determined. The goal is to detect the location of the target and evaluate the state of the target. To achieve this, the following steps which are similar to most tracking-by-detection approaches are processed sequentially. First, by applying the sliding window method to the current frame, the classifier in the competitive subset is used to classify each patch of this frame. Second, the OP method is used to compare the two targets in the two successive frames. Third, the probabilities $p(a^t|\mathbf{a}^{t-1})$ and $p(a^t|\Omega_a^t)$ are calculated. Fourth, all states classified as positive samples by $\mathbf{f}^t$ will be fused and the optimal state $\mathbf{a}^t$ with the highest confidence in the current frame will be obtained. Finally, the classifiers will be updated according to the present performance. The entire procedure is organised as in Algorithm 1. According to the results of current frame, how to search optimal classifier for next frame will be given in the following section.

Algorithm 1 LPP Tracker

Initialization Define a target in the first frame and build a classifier f_1.
Repeat $t = 1, \cdots$
(0) Capture a new frame. **If** no frame: **Exit**.
(1) Run each classifier $f_i \in \Omega_a^t$ of the competitive subset on the present frame.
(2) Combine the results $\mathbf{f}^t$ according to Eqn. 3 and obtain the best results: $\mathbf{a}^t$.
(3) **If** $\mathbf{a}^t$ is valid target: Compute the probabilities $p(a_t|\mathbf{a}_{t-1})$ and $p(a_t|f_i)$.
(4) **If** $p(\mathbf{a}^t|\mathbf{a}^{t-1}) > \tau_1$: Collect and weight samples X^t,
(5) **If** $p(\mathbf{a}^t|\Omega_a^t) > \tau_2$: Update the old classifiers f_i,
(6) **Else If** $p(\mathbf{a}^t|\Omega_c^t) > \tau_2$: Revive a classifier from Ω_c^t;
(7) **Else**: Train a new classifier $f_{n_a^t+1}$.
End
End
(8) Resampling and evaluate the classifiers.
Return Update the classifier $\mathbf{f}^{t+1}$ and set the state $\mathbf{a}^t$, **Go To** (0).

E. Collecting and Weighting Samples

If $p(\mathbf{a}^t|\mathbf{a}^{t-1}) > \tau_1$ is satisfied, it means that the tracked target is valid and can be used to update the set of classifiers. Otherwise, when no valid target is in the current frame, we can directly jump to the classifier sampling step.

1) Collecting: The sample set X^t is constructed as follows: If the overlap of $\mathbf{a}^t$ and a_l^t exceeds 0.5, the patch z_l^t of state a_l^t will be considered as the positive sample; otherwise if the overlap of $\mathbf{a}^t$ and a_l^t is lower than 0.2, it is considered as the negative sample. Also, according to the fused results $\mathbf{a}^t$, 400 positive samples will be generated by the affine warping of the selected patch $\mathbf{z}^t$ to increase the richness of positive samples.

2) Weighting: At the beginning ($t = 0$), the distribution of samples D^t used to train the first classifier is set to be equal to $1/n_X^t$. If ($t > 0$), the distribution of patches in the tth frame will be computed. Firstly, the current ensemble $\mathbf{f}^t$ is evaluated on the new patches X^t: $E^t = \frac{1}{n_X^t}\sum_{l=1}^{n_X^t} \Gamma\lfloor sign(\mathbf{f}^t(x_l^t)) \neq y_l^t \rfloor$. Secondly, sample weights D_l^t of x_l^t are defined by:

$$D_l^t = \begin{cases} E^t, & sign(\mathbf{f}^t(x_l^t)) = y_l^t; \\ 1, & otherwise. \end{cases} \tag{7}$$

Finally, set $D_l^t \Leftarrow D_l^t / \sum_{l=1}^{n_X^t} D_l^t$. Normalizing the error weights by their sum then provides us the updated penalty distribution. Samples of the new environment x_l^t, which are not recognised by the existing knowledge base $\mathbf{f}^t$, are identified.

F. Sampling the Classifiers

In this section, how to approximate the optimal classifier based on Learn++, according to the recent performance of the ensemble, will be introduced. The strategies include learning the new samples of the existing classifiers, reviving the old classifiers, training a new classifier and sampling all of them. If the current competitive subset can deal with the changes, the optimal classifier in the next frame has the same basic classifiers. To increase the adaptivity, the new samples will be learned by existing classifiers. For each $f_i \in \Omega_a^t$, if $p(\mathbf{a}^t|f_i) > \tau_2(i)$, then f_i will be updated by the samples X^t according to their distribution D_l^t following Eqn. 6. Otherwise, reviving the old classifiers or training a new classifier will be considered. This step of selecting the samples is important for keeping the stability of the classifiers. From the analysis of "concept drift", we can see that most classifiers are trained on a small dataset. If some samples generated by different distributions are used to update the classifiers, the intrinsic structure of the classifiers will be easily destroyed. Thus, the model will fail to deal with any challenges even if the classifiers have incorporated the history information.

1) Reviving: If $p(\mathbf{a}^t|\Omega_a^t) < \tau_2$ and $p(\mathbf{a}^t|\mathbf{a}^{t-1}) > \tau_1$, due to the "concept drift", it means that the current ensemble cannot deal with the changes. So, a new set of basic classifiers need to be built. New classifiers will be added into the ensemble so that the optimal classifier will be searched in a new set of classifiers. Firstly, all existing classifiers $f_i \in \Omega_c^t$ will be used to check whether the current appearance can be recognised or not by old classifiers. If a similar "concept drift" has occurred before, an old classifier can be revived. This procedure is efficient to compute because no sliding window is needed. If $p(\mathbf{a}^t|\Omega_c^t) > \tau_2$, where $p(\mathbf{a}^t|\Omega_c^t) = \max_i p(\mathbf{a}^t|f_i)$, there exists one classifier f_i that can recognise the current state. Thus, this classifier f_i will be revived directly without adding a new one. Otherwise, a new classifier will be trained and added to the ensemble following Eqn. 5.

2) Resampling: No matter whether the valid target has been detected in the current frame or not, some classifiers killed before will be revived through the resampling procedure according to the historical weights W_i^t. This will increase the diversity and avoid the local optimal solution. The adaptive rejection sampling method [47] is employed to realize this step.

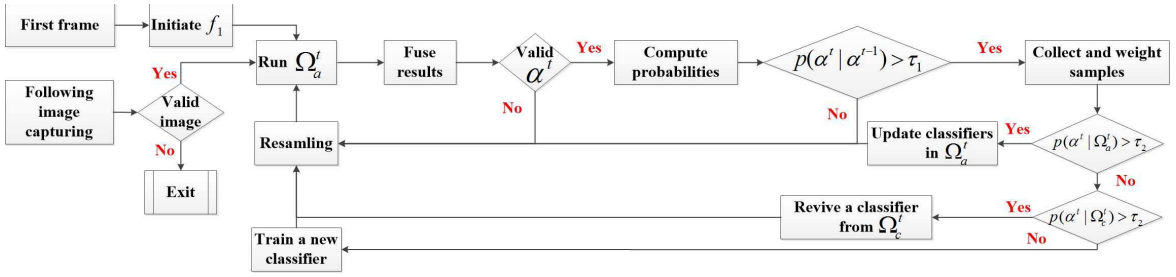


Fig. 5. The flowchart of the proposed method.

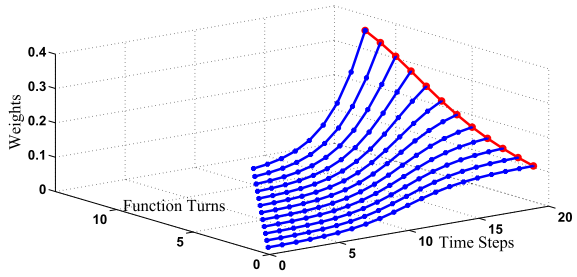


Fig. 6. Signoidal weights used in Eqn. 9. λ_1, λ_2 and η are set to 0.5, 10 and 8, respectively.

3) Evaluating: After learning, reviving or training, the competitive set of basic classifiers Ω_a^{t+1} is fixed. For finding the optimal classifier for the next frame, evaluating all classifiers $f_i \in \Omega_a^{t+1}$ on the new data X^t is necessary. Firstly, the error of each $f_i \in \Omega_a^{t+1}$ on weighting samples is defined as:

$$\varepsilon_i^t = \sum_{l=1}^{n_X^t} D_l^t \Gamma \lfloor f_i(x_l^t) \neq y_l^t \rfloor \tag{8}$$

Thus, $\varepsilon_i^t \Leftarrow \varepsilon_i^t/(1-\varepsilon_i^t)$. ε_i^t can be considered as the performance of the function. If f_i contributes mostly to the error of the ensemble classifier $\mathbf{f}^t$, ε_i^t will be larger than others. Secondly, for incorporating the performance on recent frames, a sigmoidal error weight is defined as: $\gamma_i^t(m) = 1/(1+\exp(\lambda_1\ m-\lambda_2))$, $\{m=0,\cdots,n_e^t+\eta-i\}$, where λ_1, λ_2 are two parameters, η is the time step and i is the index of the function in the ensemble. Thus, the weights are normalised so that $\gamma_i^t(m) \Leftarrow \gamma_i^t(m)/\sum_m \gamma_i^t(m)$ (see Fig. 6). Finally, the error of $f_i \in \Omega_a^{t+1}$ is weighted with respect to time so that recent competence (error rate) is considered more heavily for categorizing knowledge. The weighted errors are defined by:

$$\beta_i^t = \sum_{m=0}^{n_e^t+\eta-i} \gamma_i^t(n_e^t+\eta-i-m)\varepsilon_i^{t-m} \tag{9}$$

Thus, we calculate the classifier voting weights: $w_i^t = \log 1/\beta_i^t$ and normalise them: $\mathbf{w}_i^{t+1} \Leftarrow w_i^t/\sum_i w_i^t$. The instant voting weights can be used to update the historical weights according to $W_i^{t+1} \Leftarrow (1-\alpha)W_i^t + \alpha\mathbf{w}_i^t$, where α is the updating rate and is set to 0.05.

4) Optimal Approximation: To balance the increase of the diversity of the ensemble and efficiency of the model, the following conditions will be considered: (1) For any $f_i \in \Omega_a^{t+1}$ with $\mathbf{w}_i^{t+1} < \tau_3$, the classifiers will be killed and moved to Ω_c^{t+1}; (2) For any $f_i \in \Omega_c^{t+1}$ with $W_i^{t+1} < \tau_3$, the classifiers will be deleted for ever. Because the size of Ω_a^{t+1} is much smaller than Ω_e^{t+1}, the weights $\mathbf{w}^{t+1}$ are sparse. Therefore, the optimal approximation classifier used in the next frame will be defined by:

$$\mathbf{f}^{t+1} = \sum_{f_i \in \Omega_e^{t+1}} \mathbf{w}_i^{t+1} f_i. \tag{10}$$

VI. Experiments

In our experiments, the greyscale images are taken as input. τ_1, τ_2 and τ_3 are set to 0.75, 0.9 and 0.05, respectively. To compare with numerous methods, two types of metric are used to evaluate the different methods. (1) Center location distance: following [45], if the distance between the center of the tracked patch and the center of ground truth is within 20 pixels, the estimated target is considered as correct. Thus, the precision can be defined as the proportion of the correctly tracked frames to the total number of frames. (2) Bounding box overlap: it is defined using the ratio between the intersection and the union of the detected area and the ground truth area. Generally, if the ratio is larger than 0.5, the estimated target is considered as correct. We first compare the proposed LPP on four challenging sequences for on-way vehicle tracking and then validate it on a benchmark dataset. The flowchart of the proposed LPP is given in Fig. 5.

To demonstrate the capabilities of our system for on-road tracking, we compare the LPP tracker with five state-of-the-art methods including MEEM [48], MUSTER [49], TGPR [28], Struck [29] and TLD [23]. MEEM adopts a multi-expert scheme to address the model drift problem and the basic consideration is to avoid the model update using inconsistent samples. In MUSTER, a dual-component approach consisting of short- and long-term memory stores is proposed to process target appearance memories for tracking. Similarly, the results from two individual trackers, one derived from the auxiliary samples and the other from the target samples, are fused in the framework of TGPR. The three trackers including MEEM, MUSTER and TGPR are the ensemble-based methods. Moreover, Struck is the best method in both evaluations [30] and [45]. TLD also adopts the Fern model but the classifier members are treated equally, resulting in reduction of the model diversity.

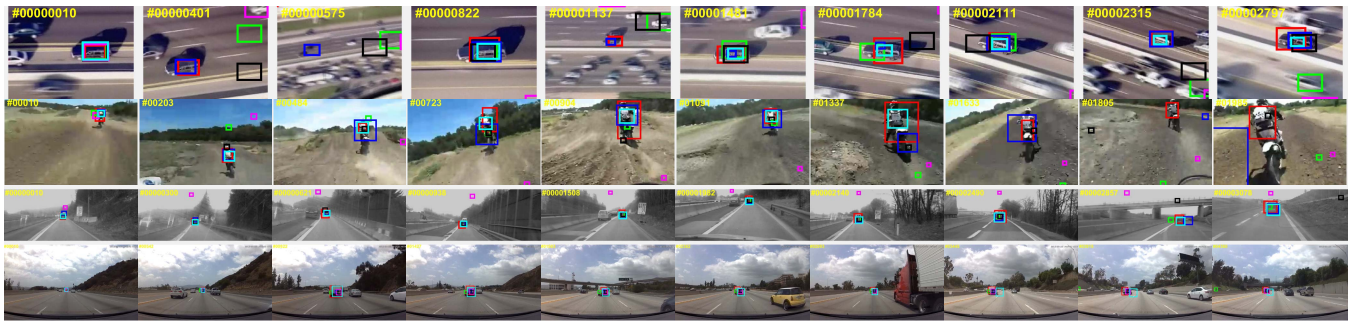


Fig. 7. Screenshots of tracking results on 4 challenging sequences including *escaping* (top row), *motocross* (second row) *suv* (third row) and *van* (bottom row). Six colors are used to distinguish the six methods: Red-LPP, Green-MEEM, Blue-MUSTER, Black-Struck, Magenta-TGPR and Cyan-TLD.

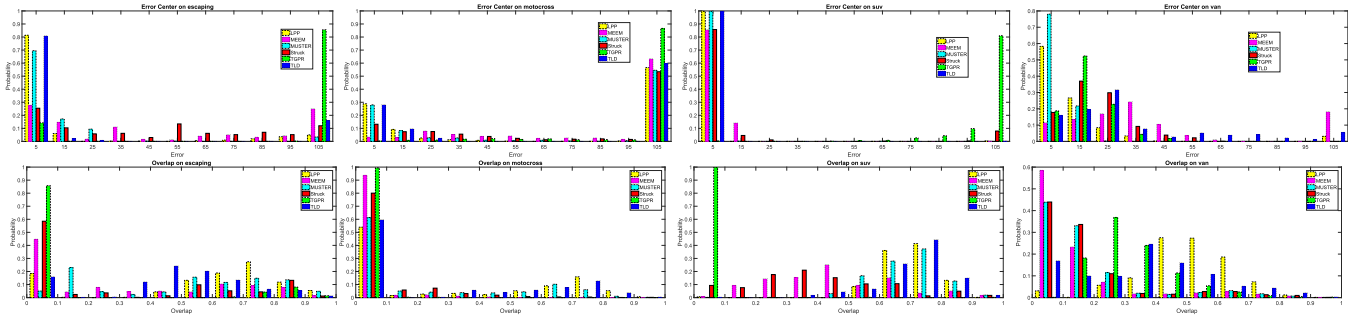


Fig. 8. The overlap and the pixel distance comparisons of 6 tracking methods on 4 challenging sequences.

TABLE II

THE COMPARISON OF EFFICIENCY BETWEEN THE 6 METHODS ON THE SEQUENCE 'VAN'. THE SIZE OF IMAGES IS 1280 × 720

Methods	LPP	MEEM	MUSTER	Struck	TGPR	TLD
FPS	25.3	21.3	1.1	8.4	0.4	2.1

TABLE III

THE DETAILS OF FOUR SEQUENCES FOR ON-WAY VEHICLE TRACKING

Sequences	van	motocross	suv	escaping
View	back	back	back	top
Ref	[17]	[23]	[30]	[30]
#frames	4500	2665	3094	2818

A. On-Road Vehicle Tracking

In this section, we choose 4 long-term sequences to validate the performance of the proposed method for vehicle tracking, including *van*, *motocross*, *suv* and *escaping* (Table III). Besides the general challenges, there are several difficulties in the selected sequences for on-road vehicle tracking, which are normally not considered by other methods before: (1) the target makes a complete rotation (*motocross*); (2) the target moves out of view and gets back with a totally different appearance and location (*escaping*); (3) the video is very long and various challenges appear simultaneously. To some extent, the assumptions of smooth motion and smooth variation necessary for most methods are not valid anymore in such sequences. The four sequences with the above three difficulties will be good examples to test the flexibility and stability of a vehicle tracking method. The screenshots of tracking results on 4 challenging sequences are given in Fig. 7. We can see that various challenges occur in the four sequences.

Firstly, we compare the efficiency of the six methods on the sequence 'van' as shown in Table II. From this table, we can see that, for a video sequence with the frame resolution of 1280 × 720, the speed of MEEM and LPP can be up to real-time performance, which is close to the speed of image capturing (30FPS) on a normal platform of Dell M4600 (Intel Core 2:8 GHz and 8G RAM). However, later, we can see that the tracking performance of MEEM for on-road tracking is much worse than that of LPP.

Secondly, the statistical comparison of centre distances between the tracked patch and the ground truth patch is given in the first row of Fig. 8. To better illustrate the performance, all distances are grouped into 11 bins and the statistics are given for each bin. For the distance which is less than 100 pixels, we divide them into 10 equal bins but all the distances that are larger than 100 pixels are considered in one group. Actually, almost all methods could not track the target when the distances are larger than 100 pixels. From this figure, we can see that the error percentage close to 0 of LPP is higher than that of other methods and we can conclude the tracking performance of LPP is better than that of others. The corresponding precision defined using centre error is given in Table IV. Except for MUSTER which will be discussed later in the criterion of the bounding box overlap, LPP achieves better results than other methods on the four sequences. For the sequence *motocross*, it is worth to point out that the predefined target is not present in the camera view after 2035 frames thus we just report the results before the 2035th frame.

TABLE IV

THE PERCENTAGES OF FRAMES IN WHICH THE PIXEL DISTANCE OF CENTRES BETWEEN THE DETECTED AREA AND GROUNDTRUTH IS SMALLER THAN 20

Sequences	van	motocross	suv	escaping
LPP	0.849	0.380	0.997	0.874
MEEM	0.250	0.063	0.997	0.423
MUSTER	0.998	0.363	0.998	0.868
Struck	0.547	0.206	0.903	0.359
TGPR	0.711	0.011	0.003	0.142
TLD	0.355	0.376	0.997	0.832

TABLE V

THE PERCENTAGES OF FRAMES IN WHICH THE RATE OF OVERLAP AREA BETWEEN THE DETECTED LOCATION AND GROUNDTRUTH TO THEIR COMBINATION AREA IS LARGER THAN 0.5

Sequences	van	motocross	suv	escaping
LPP	0.543	0.364	0.993	0.768
MEEM	0.075	0.004	0.347	0.335
MUSTER	0.081	0.220	0.966	0.605
Struck	0.080	0.020	0.292	0.339
TGPR	0.095	0.000	0.003	0.142
TLD	0.228	0.304	0.936	0.469

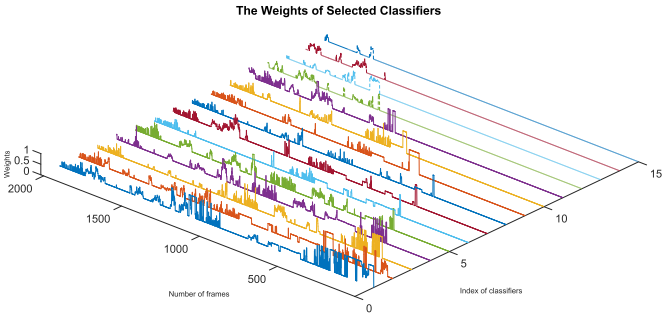


Fig. 9. The weights for the optimal classifier $\mathbf{f}^t$. Due to the absent of the vehicle in the last frames, the weights are shown only before 2035 frames.

Thirdly, the statistical comparison of bounding box overlaps between the tracked patch and the ground truth patch is given in the second row of Fig. 8. Obviously, the higher the overlap, the better results are achieved by one method. We can see that the overlap ratios of LPP for most of frames are much closer to 1 than that of other methods. This demonstrates that better results are achieved by LPP. The corresponding precision defined using bounding box overlap is given in Table V. From this table, we can see that LPP distinctively outperforms all other methods on the four sequences. In fact, using the criterion of centre distance, MUSTER performs much better than other methods on sequences *van* and *suv*, but, using the criterion of bounding box overlap, its results are not as good. This is because that MUSTER could catch small sub-parts of predefined target thus, for some frames, the centre distances are low whist the overlap ratios are very low. In contrast, LPP can track the locations of the predefined target and estimate the scale states more robustly.

Finally, Fig. 9 demonstrates the weights of classifiers on all the frames of sequence *motocross*. When no valid target is detected, LPP tracker will sample the classifiers according to their historical weights. Once the predefined target appears in view, LPP tracker will select the most effective classifier to track the target. From Fig. 9, we can see that the weights are very sparse and just a few members (less than five) will be run for each frame but totally 15 classifiers could be maintained.

B. Comparison With the Representative Methods

Actually, besides the on-road vehicle tracking, other general moving objects on the road are still the critical factors of the vehicle safety and are necessary to be detected and tracked for risk identification and prediction. To further evaluate the proposed method for general object tracking, in this section, LPP tracker will be compared with 13 representative methods, including Struck [29], SCM [19], VTS [35], VTD [50], CSK [51], ASLA [18], DFT [52], L1APG [53], LSK [54], MIL [4], OAB [31], Frag [12] and CT [22]. Most of them were recently proposed and ranked in the top list in the experimental comparison in [45]. Our experiments follow the setting in [45]. Each sequence is repeated 5 times with different random seeds by LPP tracker, and the median results are reported.

The success plot shows the ratios of successful frames at the overlap thresholds varying from 0 to 1 while the precision plot shows the ratios of successful frames at the centre location error varying from 0 to 50 pixels. The area under curve (AUC) [45] of each plot is used to rank the tracking algorithms. Wu *et al.* [45] pointed out that AUC is fairer and more accurate because it measures the overall performance. Both success plots and AUC rankings are shown in Fig. 10. The precision rankings of the 14 methods on the 51 videos for 11 types of challenges are given in Table VI.

Firstly, from Fig. 10, LPP tracker can achieve the best performance among all the 13 methods (Only top 10 are shown in the figure) on most of the challenges, from the perspectives of both center location distance and bounding box overlap. From this figure, we can see that LPP outperforms other methods obviously both at the overlap range from 0 to 0.6 and the location distance range from 15 to 50. Generally, sparse representation based methods including SCM and ASLA achieve better results both at higher overlap rates and nearer pixel distances. In fact, at this situation, LPP tracker performs very closely to the best method and the performances of all methods are nearly the same.

Secondly, the performance comparisons on different challenges are given in Table VI. On the one hand, using the bounding box overlap criterion, LPP tracker has a great advantage over the other methods except for the challenges of scale variation and low resolution. Although LPP tracker is not the best method for these two challenges, it still ranks in the top three. It is also meaningful to point out that LPP tracker achieves much better results on the challenges of fast motion, motion blur and out of view than other methods. In fact, the first motivation of LPP tracker is to address the challenges of appearance changes during movement. For example, by updating the selected classifiers, the "concept" can be reflected by the independent particular classifier. If the object leaves out of view then comes back, LPP tracker can still track it using a particular classifier as long as the appearance

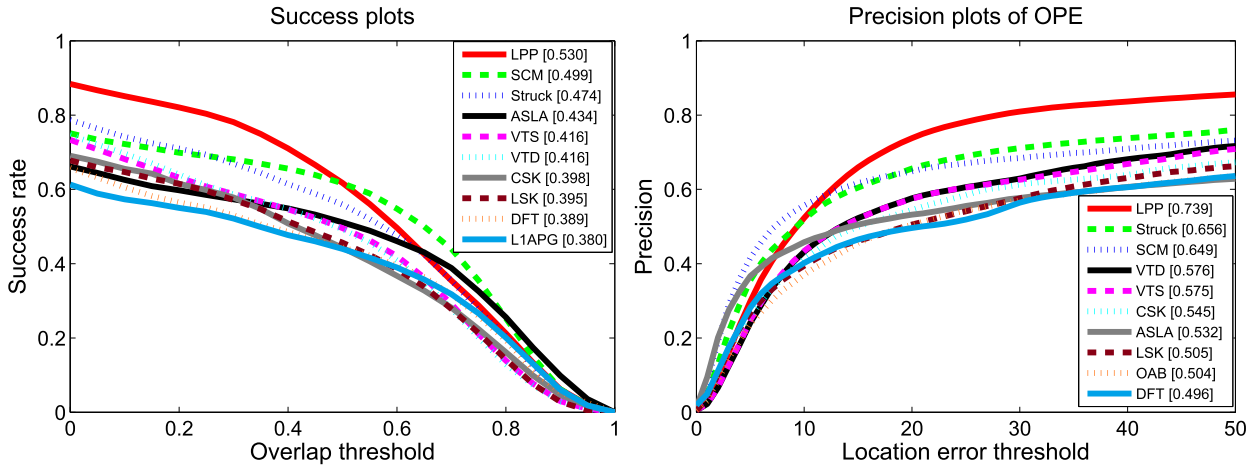


Fig. 10. The success and precision plots and AUC rankings of 10 tracking methods.

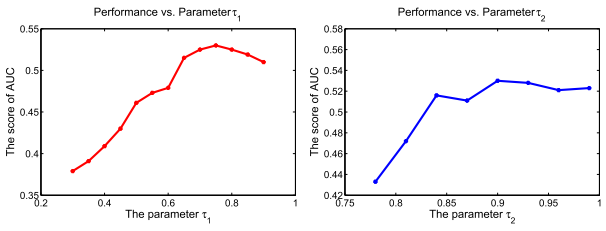
TABLE VI
THE PRECISION RANKINGS OF 14 TRACKING METHODS ON CHALLENGING SEQUENCES

Challenges	LPP	Struck	SCM	ASLA	CSK	L1APG	OAB	VTD	VTS	DFT	LSK	CT	MIL	Frag
IV	0.677	0.558	0.594	0.517	0.481	0.341	0.388	0.557	0.573	0.475	0.449	0.359	0.349	0.326
OPR	0.725	0.597	0.618	0.518	0.540	0.478	0.503	0.620	0.604	0.497	0.525	0.394	0.466	0.444
SV	0.720	0.639	0.672	0.552	0.503	0.472	0.541	0.597	0.582	0.441	0.480	0.448	0.471	0.407
OCC	0.719	0.564	0.640	0.460	0.500	0.461	0.483	0.545	0.534	0.481	0.534	0.412	0.427	0.475
DEF	0.734	0.521	0.586	0.445	0.476	0.383	0.470	0.501	0.487	0.537	0.481	0.435	0.455	0.468
MB	0.663	0.551	0.339	0.278	0.342	0.375	0.360	0.375	0.375	0.383	0.324	0.306	0.357	0.288
FM	0.679	0.604	0.333	0.253	0.381	0.365	0.416	0.352	0.353	0.373	0.375	0.323	0.396	0.346
IPR	0.677	0.617	0.597	0.511	0.547	0.518	0.471	0.599	0.579	0.469	0.534	0.356	0.453	0.401
OV	0.698	0.539	0.429	0.333	0.379	0.329	0.454	0.462	0.455	0.391	0.515	0.336	0.393	0.355
BC	0.693	0.585	0.578	0.496	0.585	0.425	0.446	0.571	0.578	0.507	0.504	0.339	0.456	0.421
LR	0.501	0.545	0.305	0.156	0.411	0.460	0.376	0.168	0.187	0.211	0.304	0.152	0.171	0.163

has been learned in a certain time. On the other hand, using the center location distance criterion, LPP tracker outperforms all other methods on most challenges only except for the low resolution challenge. The advantages using pixel distance criterion are more obvious than the overlap rate criterion. Unlike template matching methods, in some difficult situations including out-of-plane rotation and occlusion, LPP tracker can track the part of object without containing the pixels from environment. However, at such conditions, the ground truth annotations of video samples in the dataset normally contain the pixels from the environment.

In total, LPP tracker gains nine firsts, one second and one third by the AUC ranking, and it gains ten firsts and one second by the precision ranking. The difference between the two rankings is scale variation. That is because LPP tracker can build a new classifier for one part of the object when there are some large deformations in the remaining part in these challenges, where the object has been tracked by LPP tracker but the score of overlap is relatively low.

Finally, there are two parameters of motion constraints τ_1 and τ_2 to guide the learning of LPP tracker. When we investigate one parameter, other parameters will be set to default (same values for all videos). In Fig. 11 (a) and (b), the overall performance on all the videos vs. the different settings for the two parameters are given. We can see that the parameter τ_1 achieves the best performance around 0.75 while



(a) (b)

Fig. 11. (a) The AUC performance vs. parameter τ_1. (b) The AUC performance vs. parameter τ_2.

the parameter τ_2 achieves the best performance around 0.9. If the two parameters are set too small, the model will become more flexible but less stable. More erroneous information will be added into the model and the performance will deteriorate. However, if the two parameters are set too large, the model cannot adapt to the new environment and the performance of the model will decrease as well. Moreover, the scores of AUC are relatively stable around the best values of the two parameters, which means they are not very sensitive.

VII. CONCLUSION

In this paper, we have proposed a long-term motion tracker for the intelligent vehicles and general objects. By means of

automatically adjusting the members of classifiers, a democracy mechanism is adopted to solve numerous challenges appearing around the vehicles, simultaneously. Moreover, the proposed tracker achieves an optimal balance between flexibility and stability of the classifiers and between the efficiency and performance of the model as well. Essentially, the rapid development of hardware on intelligent vehicles makes it feasible to explore more complex models for motion tracking. In future work, on the one hand, combining different successful models to further improve the diversity of the tracker is meaningful. On the other hand, to track an object more robustly, a system of multi-cameras could be used as well.

REFERENCES

[1] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 4, pp. 1773–1795, Dec. 2013.

[2] X. Gibert, V. M. Patel, and R. Chellappa, "Deep multitask learning for railway track inspection," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 1, pp. 153–164, Jan. 2017.

[3] Y. Yuan, Z. Xiong, and Q. Wang, "An incremental framework for video-based traffic sign detection, tracking, and recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 7, pp. 1918–1929, Jul. 2016.

[4] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619–1632, Aug. 2011.

[5] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 125–141, 2008.

[6] R. T. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1631–1643, Oct. 2005.

[7] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.

[8] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof, "PROST: Parallel robust online simple tracking," in *Proc. CVPR*, Jun. 2010, pp. 723–730.

[9] F. Zheng, L. Shao, J. Brownjohn, and V. Racic, "Learn++ for robust object tracking," in *Proc. BMVC*, 2014.

[10] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.

[11] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. IJCAI*, 1981, pp. 674–679.

[12] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proc. CVPR*, Jun. 2006, pp. 798–805.

[13] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. CVPR*, Jun. 2001, pp. 142–149.

[14] Q. Wang, P. Yan, Y. Yuan, and X. Li, "Multi-spectral saliency detection," *Pattern Recognit. Lett.*, vol. 34, no. 1, pp. 34–41, 2013.

[15] J. Kwon and K. M. Lee, "Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping Monte Carlo sampling," in *Proc. CVPR*, Jun. 2009, pp. 1208–1215.

[16] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[17] A. Li, M. Lin, Y. Wu, M.-H. Yang, and S. Yan, "NUS-PRO: A new visual tracking challenge," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 335–349, Feb. 2015.

[18] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. CVPR*, Jun. 2012, pp. 1822–1829.

[19] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Proc. CVPR*, Jun. 2012, pp. 1838–1845.

[20] S. Zhang, H. Yao, X. Sun, and X. Lu, "Sparse coding based visual tracking: Review and experimental comparison," *Pattern Recognit.*, vol. 46, no. 7, pp. 1772–1788, 2013.

[21] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, Aug. 2004.

[22] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Proc. ECCV*, 2012, pp. 864–877.

[23] Z. Kalal, J. Matas, and K. Mikolajczyk, "P-N learning: Bootstrapping binary classifiers by structural constraints," in *Proc. CVPR*, Jun. 2010, pp. 49–56.

[24] F. Tang, S. Brennan, Q. Zhao, and H. Tao, "Co-tracking using semi-supervised support vector machines," in *Proc. CVPR*, Oct. 2007, pp. 1–8.

[25] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *Proc. ECCV*, 2008, pp. 234–247.

[26] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. van den Hengel, "Robust tracking with weighted online structured learning," in *Proc. ECCV*, 2012, pp. 158–172.

[27] G. Li, L. Qin, Q. Huang, J. Pang, and S. Jiang, "Treat samples differently: Object tracking with semi-supervised online covboost," in *Proc. ICCV*, Nov. 2011, pp. 627–634.

[28] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with Gaussian processes regression," in *Proc. ECCV*, 2014, pp. 188–203.

[29] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *Proc. ICCV*, Nov. 2011, pp. 263–270.

[30] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.

[31] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. CVPR*, Jun. 2006, pp. 260–267.

[32] X. Yan, X. Wu, I. A. Kakadiaris, and S. K. Shah, "To track or to detect? An ensemble framework for optimal selection," in *Proc. ECCV*, 2012, pp. 594–607.

[33] Q. Wang, J. Fang, and Y. Yuan, "Multi-cue based tracking," *Neurocomputing*, vol. 131, pp. 227–236, May 2013.

[34] Y. Yuan, J. Fang, and Q. Wang, "Robust superpixel tracking via depth fusion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 15–26, Jan. 2013.

[35] J. Kwon and K. M. Lee, "Tracking by sampling trackers," in *Proc. ICCV*, Nov. 2011, pp. 1195–1202.

[36] M. Isard and A. Blake, "Icondensation: Unifying low-level and high-level tracking in a stochastic framework," in *Proc. ECCV*, 1998, pp. 893–908.

[37] J. H. Yoon, M.-H. Yang, and K.-J. Yoon, "Interacting multiview tracker," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 5, pp. 903–917, May 2015.

[38] Q. Bai, Z. Wu, S. Sclaroff, M. Betke, and C. Monnier, "Randomized ensemble tracking," in *Proc. ICCV*, Dec. 2013, pp. 2040–2047.

[39] J. Xing, J. Gao, B. Li, W. Hu, and S. Yan, "Robust object tracking with online multi-lifespan dictionary learning," in *Proc. CVPR*, Dec. 2013, pp. 665–672.

[40] S. He, Q. Yang, R. W. H. Lau, J. Wang, and M.-H. Yang, "Visual tracking via locality sensitive histograms," in *Proc. CVPR*, Jun. 2013, pp. 2427–2434.

[41] X. Sun, N. H. C. Yung, and E. Y. Lam, "Unsupervised tracking with the doubly stochastic Dirichlet process mixture model," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 9, pp. 2594–2599, Sep. 2016.

[42] D. Achlioptas, "Database-friendly random projections: Johnson–Lindenstrauss with binary coins," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 671–687, 2003.

[43] M. Ozuysal, P. Fua, and V. Lepetit, "Fast keypoint recognition in ten lines of code," in *Proc. CVPR*, Jun. 2007, pp. 1–8.

[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[45] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. CVPR*, Jun. 2013, pp. 2411–2418.

[46] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. CVPR*, Jun. 2005, pp. 886–893.

[47] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2007.

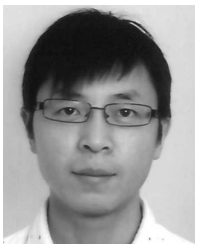
[48] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: Robust tracking via multiple experts using entropy minimization," in *Proc. ECCV*, 2014, pp. 188–203.

[49] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao, "MUlti-store tracker (MUSTer): A cognitive psychology inspired approach to object tracking," in *Proc. CVPR*, Jun. 2015, pp. 749–758.

[50] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Proc. CVPR*, Jun. 2010, pp. 1269–1276.

[51] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. ECCV*, 2012, pp. 702–715 .
[52] L. Sevilla-Lara and E. Learned-Miller, "Distribution fields for tracking," in *Proc. CVPR*, Jun. 2012, pp. 1910–1917.
[53] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust ℓ_1 tracker using accelerated proximal gradient approach," in *Proc. CVPR*, Jun. 2012, pp. 1830–1837.
[54] B. Liu, J. Huang, L. Yang, and C. Kulikowsk, "Robust tracking using local sparse appearance model and k-selection," in *Proc. CVPR*, Jun. 2011, pp. 1313–1320.

Feng Zheng is currently pursuing the Ph.D. degree with the Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield, U.K. His research interests include computer vision, machine learning, and human–computer interaction.



Ling Shao (M'09–SM'10) is currently a Professor with the School of Computing Sciences, University of East Anglia, U.K., and a Visiting Professor with Northwestern Polytechnical University, China. His research interests include computer vision, machine learning, and multimedia.

Junwei Han is currently a Professor with Northwestern Polytechnical University, Xi'an, China. His research interests include computer vision and brain imaging analysis.