MIL based Visual Object Tracking with Kernel and Scale Adaptation

Vijay K. Sharma, K.K. Mahapatra

www.elsevier.com/locate/image

PII: S0923-5965(17)30008-5
DOI: http://dx.doi.org/10.1016/j.image.2017.01.007
Reference: IMAGE15168

To appear in: *Signal Processing : Image Communication*

Received date: 13 July 2016
Revised date: 21 January 2017
Accepted date: 23 January 2017

Cite this article as: Vijay K. Sharma and K.K. Mahapatra, MIL based Visua Object Tracking with Kernel and Scale Adaptation, *Signal Processing : Imag Communication,* http://dx.doi.org/10.1016/j.image.2017.01.007

MIL based Visual Object Tracking with Kernel and Scale Adaptation

Vijay K. Sharma and K. K. Mahapatra

***Abstract*—Multiple instance learning (MIL) is a framework wherein training examples are provided in form of labeled bags rather than labeled instances. For visual object tracking, the MIL framework is used to select a few discriminative features from a pool of features by maximizing the bag likelihood. The feature pool consists of computationally efficient Haar-like features. Each feature is used for the construction of weak classifier also called decision stump. A few classifiers with high discriminative power (to separate target from background) are combined to form a strong classifier. The key point is the selection of a few highly discriminative features from a very large pool in small amount of time to ensure fast and accurate tracking. In this paper, we propose a fast online feature selection algorithm based on maximizing the classifier score (CSR). The tracking performance is better than the existing feature selection methods. This paper also introduces kernel trick on Haar-like features for target tracking. Furthermore, we explore use of Haar-features in half target space. To get the actual target size in successive video frames, a scaling strategy is applied. By having the kernel correlation, Haar-features in half target spaces and scale adaptation method in a single tracking framework, we obtain better tracking performance as compared to the state-of-the-art trackers.**

***Index Terms*—Computer vision, visual tracking, multiple instance learning, weak classifier, classifier score, Haar-features, kernels.**

I. Introduction

VISUAL object tracking finds many practical applications in computer vision that include human computer interaction, guidance of vehicles in automated vehicle control systems, etc [1], [2]. The task involved in object tracking is the estimation of trajectory in the successive video frames. The main challenges that lie in the tracking are appearance change from frame to frame, presence of features in the background similar to that of object (clutter), partial occlusion of the target object, change of scene illumination, and need for real-time processing. The tracking algorithm should have speed high enough to be able to track the target from a real-time video. For example, hand tracking in gesture identification [3], tracking of objects for visual surveillance, tracking of a missile or tracking in robot vision [4] must be performed in real-time [5].

The idea of tracking can be realized as a binary classification problem wherein target object is separated from the background [6]. Selecting a small number of features and constructing a simple classifier from each feature can make the process of classification faster as demonstrated in [7], where simple classifiers are built using Haar-like features by AdaBoost algorithm for face detection.

Babenko et al.[8] state that training the appearance model that is adaptive is a difficult task and therefore used multiple instance learning (MIL) framework to handle the ambiguities in the training samples. In MIL, training examples come in bags. In a bag labeled positive, at least one example is positive while in negative bag all examples are negative [9], [10]. The formulation in [11] makes assumption that the positive bags can be modeled as the mixture distribution containing concept and non-concept and instances are drawn from this distribution. For tracking, positive examples are taken around the small neighbor of the current target and negative examples are at some distance away from the target center. Choosing multiple instances for positive example prevents the problem of updating appearance model with a sub optimal one when the tracking accuracy is low. All positive instances are kept in positive bag and all the negative instances are kept in negative bag. The bagging of examples handles the ambiguity in the training data.

In MIL framework, MILBoost algorithm is used in [9] for training the classifier. Online MILBoost in [8] is a feature selection algorithm based on MIL framework. It is a modification of MILBoost and online AdaBoost in [12]. In online MILBoost, the bag probability is modeled as Noisy-OR (NOR), similar to MILBoost in [9]. The online MILBoost algorithm for feature selection in [8] (simply referred as MIL in this paper) maintains a weak classifier pool and greedily selects some most discriminative weak classifiers by maximizing the log likelihood of bags. Then, set of chosen weak classifiers are combined to form a strong classifier. This strong classifier is used to classify samples which are cropped from the next video frames. The tracking location corresponds to the sample which has the maximum score. Each classifier is a weak classifier constructed by computationally efficient Haar-like feature. Therefore, choosing weak classifier is equivalent to choosing the features.

Weighted MIL (WMIL), a faster version of MIL algorithm, is proposed in [13] that gives larger weight to instances close to the correct object position and replaces NOR bag probability model with a new bag probability function. WMIL algorithm does not compute instance and bag probability M (the number of weak classifiers in pool) times after selection of one weak classifier, which results in reducing the computational complexity of the algorithm. Therefore, real-time tracking performance has been obtained in WMIL. Zhang et al. in [14] proposed online discriminative feature selection (ODFS)

Vijay K. Sharma is with the Department of Electronics and Communication Engineering, National Institute of Technology, Rourkela, Odisha, 769008 India e-mail: vijay4247@gmail.com

K. K. Mahapatra is with the Department of Electronics and Communication Engineering, National Institute of Technology, Rourkela, Odisha, 769008 India e-mail: kkm@nitrkl.ac.in

based on maximization of confidences of target samples while suppressing the confidences of background (negative) samples, i.e., they maximize the average confidence between the positive samples and negative samples. They achieved better timing performance as compared to WMIL in feature selection as observed by executing both WMIL and ODFS algorithms on our system.

In this paper, we propose a method for online feature selection in multiple instance learning (MIL) framework based on maximizing the classifier functional value. We treat a classifier corresponding to a feature in positive instance as independent with respect to classifier corresponding to the same feature in negative instances and state that classifiers which have very high discriminative power should have high functional magnitude from positive and negative examples combined in their relative discrimination power in the pool. There is no sigmoid function in proposed feature selection method unlike other MIL based feature selection methods. We call the proposed algorithm as CSR (stands for Classifier ScoRe) based feature selection. It is comparatively faster than both ODFS and WMIL based feature selection algorithms. The tracking accuracy of the MIL based tracker is lower than some other state-of-the-art trackers. This can be because of inefficient method for updating mean and variance of classifiers.

To improve tracking accuracy, we introduce 8 different variants in CSR. Two intermediate tracking instances are cropped using ratio classifiers, while the two more intermediate tracking instances are cropped using feature similarity computed by kernel trick on Haar-features. Four additional tracking instances are cropped using ratio classifiers corresponding to feature subset residing in top half, bottom half, left half and right half of the target space. Scale adaption has also been introduced to get the actual size of the target in successive frames. To realize the scaling, following strategy is used. Around center of all 8 tracked instances, 4 more instances are cropped. The first 2 with increased width and height while the other 2 with decreased width and height. Altogether, $8 \times 5 = 40$ instances are cropped and compared for their correlation with the updated target. The instance which has highest correlation is selected as the final estimated target sample. We call the algorithm as CSRKS (CSR with kernels and scale adaptation). The tracking accuracy of CSRKS is better than the existing state-of-the-art trackers in a number of video sequences.

The following are the main contributions of the paper.

1) For feature selection, we represent discriminative classifier constructed using a Haar-like feature from a sample in positive bag (positive instance) as independent with respect to classifier from the same Haar-like feature in sample of negative bag. While finding the resultant classifier score, more weights have been given to classifiers from the positive instance to reduce the chances of drift.
2) The computational complexity of the feature selection method is very low as compared to the existing methods.
3) Our feature selection algorithm does not contain the sigmoid function as in existing methods, and therefore it is suitable for dedicated hardware implementation for tracking in high frame rate videos.
4) We have used kernel trick on Haar-like features to find the similarity using inner product and showed that tracking performance can be improved by using similarity of two Haar-like features.
5) The feature similarity is computed between features of tracked target in previous video frame and features of the target candidates in current video frame. This step prevents the tracker from drifting solely because of non-optimal update model of classifier parameters.

All three algorithms, WMIL, ODFS and CSR (ours) use Haar-like features to form a discriminative classifier which is a weak classifier. It requires many classifiers to form a strong one. The number of features (and hence classifiers) required for forming a strong classifier in object tracking may depend on the number of rectangles used for a feature computation. However, the actual number of features for a task can be decided by the experimentation as lesser number may result in weak classifier. At the same time higher number will also turn out to be a bad classifier model because of overfitting.

This paper is organized as follows. We discuss the related works in section II. Section III describes in brief the object tracking in MIL framework and feature selection methods. We discuss our proposed feature selection algorithm in section IV. Target tracking using discriminative classifiers, Haar-like features on kernel and scale adaptation is introduced in section V. Experimental results with comparative performance analysis is given in section VI. Section VII concludes the paper.

II. Related Works

The object appearance model can be broadly classified into two categories, generative and discriminative. Based on the appearance model used, a tracker can be classified as generative, discriminative and hybrid generative discriminative.

A. Generative Trackers

A generative tracker depends only on encoding the target appearance [15] and does not contain the background information. The main aim is to accurately fit the data [16]. Kernel based object tracking (mean shift) is one of the most popular scheme which models the target using color histogram [17]. It finds the estimated target location using iterative procedure where the estimated location moves towards the maximum density in each iteration. The most important thing of mean-shift based tracking is its robustness towards scaling, rotation and partial occlusion. However, its performance efficiency degrades when there is dramatic change in intensity or color in the neighborhood. In addition, the traditional mean-shift based tracker can produce local optimal solution [18]. Iterative particle filter for visual object tracking is proposed in [18] for finding the true target (avoiding the local optimal) state by iterative sampling of the particles and contracting the search scope in each iteration for increasing the sampling density. The tracking accuracy can be higher at higher sampling density when the appearance model is also robust. For handling the abrupt motion and longtime occlusion, Li et al. in [19] integrated salient regions detected by a visual attention model into particle filter framework. Visual attention is an important

process which enables human being to select a relevant portion within a large visual data. Khan et al. in [20] estimated object appearance online on Grassmann manifolds which are described by basis matrices. The speed of this method is very low as it takes approximately 11 seconds in tracking a single video frame. Eigenbasis vectors as the appearance model has been used in [21]. The appearance model is incrementally updated, as the new training data are available, using an efficient algorithm based on singular value decomposition (SVD). This algorithm relies heavily on appearance model as the learning framework.

Appearance modeling through sparse representation in particle filter based framework is a recent research in visual tracking [22],[23],[24],[25]. The idea is to represent a sample via linear combination of some target templates and trivial templates. The solution of L-1 regularized least squares program gives the sparse coefficients for each templates [24]. The occlusion in the target candidate is represented by coefficients of trivial templates. In [25], an online local sparse dictionary learning formulation is used to capture the local target appearances. The solution of L-1 minimization program is required for each target candidates, and therefore, the computational complexity of the sparse based methods are very high. In [23], a fast solution to L-1 norm minimization based on the accelerated proximal gradient approach is introduced.

B. Discriminative Trackers

The appearance of a discriminative tracker constitutes features from both the target as well as the background. Collins and Liu [26] described that the best features should have very high discriminative power between target object and the background. They used the likelihood ratio between probability density of pixels in the object and the background to distinguish the pixels in the object from the background. Liang et al. [27] ranked features according to Bayes error rate [28] and selected top N features having smallest Bayes error rate. The features used are color feature set as in [26]. The performance of these algorithms are tested only on a few challenging video sequences. Avidan in [29] used AdaBoost to form a strong classifier from weak classifiers. The strong classifier classifies pixels in the next frame to produce the confidence map. The mean-shift algorithm finally finds the peak in the map as the object position. One of the drawbacks of this method is that the scene changes is adapted at the cost of drift in the target.

Hare et al. [30] used kernelized structured output support vector machine (SVM) framework for adaptive object tracking. The method is also called as STRUCK, whose computational complexity grows as the number of support vectors. To reduce the number of support vectors, they used a budgeting mechanism during tracking to enhance the tracking speed. The drawback of this algorithm is that with lower number of support vectors, its accuracy is affected. Kalal et al. in [31] developed a framework which integrates the tracking, detection and learning (TLD). A tracker and a detector localizes the object. An integrator combines the bounding boxes of tracker and detector. There is a tracking failure if neither detector nor the tracker outputs a bounding box. Henriques et al. in [32] proposed high speed kernelized correlation filter (KCF) based tracker. Using Fourier domain formulation, they have exploited the circulant data matrix concept to reduce the computation in training the classifier with positive and negative examples. The negative examples are nothing but all possible cyclic shifts of a base sample. Tracking by detection method in [33] is based on learning spatial context model between object and its immediate local surrounding. The algorithm presented by Yang et al. [34] is based on finding superpixels, and then forming a discriminative appearance model by using mean shift clustering.

C. Hybrid Generative Discriminative Trackers

The hybrid appearance is constructed using both generative and discriminative (containing background information) models. Xie et al. in [15] constructed an over-complete discriminative dictionary. It uses local image patches of both target and background represented by their respective sparse codes. However, this method has very low tracking speed (0.1 frames per second). Yin et al. in [35] used structured SVM (discriminative) as well as the incremental PCA (generative) to obtain a robust tracker with less target drift. Local cosine similarity is used in [36] for target and target candidates (a generative approach). After obtaining the optimal state, it learns discriminative weights using positive and negative target samples (a discriminative approach) via quadratic programming to improve the discriminative ability of the tracker.

Multi-target visual tracking is another related computer vision research in which the task is to locate and identify multiple moving objects from successive video frames [37], [38].

III. MIL Framework For Visual Object Tracking

A. Positive and Negative Example Selection

Positive examples are cropped around the target location within some radius . Let $\mathbf{x}_0$ be the target patch in the current frame (t). Positive samples are generated such that for any image patch $\mathbf{x}$, $\|\mathrm{I_t}(\mathbf{x}) - \mathrm{I_t}(\mathbf{x}_0)\| < \alpha$ where, $\mathrm{I_t}(\mathbf{x})$ represents location of a patch (sample) $\mathbf{x}$ in frame (t). Similarly, negative examples are cropped from the background near the target location such that $\gamma < \|\mathrm{I_t}(\mathbf{x}) - \mathrm{I_t}(\mathbf{x}_0)\| < \beta$ where, $\alpha < \gamma < \beta$. The number of negative samples is limited by choosing a threshold; otherwise a large number of negative samples will be generated increasing the complexity in training the classifier. The positive samples are kept in a bag labeled positive and negative samples are kept in negative labeled bag [8].

B. Feature Extraction and Appearance Model

For each samples in positive and negative bags, Haar-like features are extracted. The reason behind working on feature based system instead of pixel directly is that the former is faster to operate [39]. For calculation of Haar-like features, easily computable intermediate image representation, called integral image, is used. The discriminative appearance model

is weak classifiers, each composed of Haar-like feature f_k. The classifiers model is given by [13],

$$h_k(\mathbf{x}) = \ln\left(\frac{p_t(f_k(\mathbf{x})|\, y=1)}{p_t(f_k(\mathbf{x})|\, y=0)}\right) \tag{1a}$$

$$H_K(\mathbf{x}) = \sum_{k=1}^{K} h_k(\mathbf{x}) \tag{1b}$$

where, H_K represents strong classifier from combination of K classifiers, $\mathbf{x}$ is a sample and $f_k(\mathbf{x})$ represents k_{th} feature inside the sample $\mathbf{x}$, (y = 1) stands for positive bag whereas, (y = 0) stands for negative bag. Features probabilities are normally distributed with four parameters ($\mu_1, \sigma_1, \mu_0, \sigma_0$), i.e., $p_t(f_k(\mathbf{x})|\, y = 1) \sim \mathcal{N}(\mu_1, \sigma_1)$ and $p_t(f_k(\mathbf{x})|\, y = 0) \sim \mathcal{N}(\mu_0, \sigma_0)$

C. Online Multiple Instance Boosting

The purpose of boosting is to select a few features from a large set of feature pool to make the classification process computationally efficient [8]. Each weak classifier from Haar-like feature is a decision stump. Boosting chooses features, with most discriminative power of classifiers, by satisfying some objective function in an iterative way, i.e., one feature per iteration. The online MILBoost algorithm proposed in [8] selects K number of features (very small) by maximizing the log-likelihood of bags. The bag probability model is given by

$$p_i = 1 - \prod_j (1 - p_{ij}) \tag{2}$$

where, i is index of bag and j is index of instance. The instance probability is modeled by sigmoid function

$$p_{ij} = \sigma(H(\mathbf{x}_{ij})) \tag{3}$$

At each iteration step in boosting, a strong classifier is obtained by combination of previously selected classifiers, i.e., if $H_{\mathrm{k}-1}$ is a strong classifier made by combination of $\mathrm{k}-1$ weak classifiers then k_{th} classifier (h_k) is given by

$$h_k = \underset{h(\in h_i|i=1:M)}{\arg\max}\ \ell(H_{k-1} + h) \tag{4}$$

where, M is total number of features in feature pool. The algorithm in [8] needs to compute instance probability and bag probability M times for each weak classifier selection resulting in slower speed. There is a faster version of this algorithm given by Zhang et al. in [13] called weighted MIL boost (WMIL) that does not require M iterations for a feature selection. Online feature selection algorithm ODFS in [14] is faster than WMIL. If $c(\mathbf{x})$ be the confidence map function for an instance $\mathbf{x}$, then

$$c(\mathbf{x}) = P(\, y = 1|\, \mathbf{x}) = \sigma(H_K(\mathbf{x})) \tag{5}$$

Object location $\mathbf{x}^*$ (tracked sample) is determined by peak of the map ($\mathbf{x}^* = \mathrm{argmax}_{\mathbf{x}}\mathrm{c}\,(\mathbf{x})$).

IV. Proposed Online Feature Selection in MIL Framework

Let, N is the number of positive instances in the positive bag (X^+) and L is the number of negative instances in the negative bag (X^-). A classifier will have high discriminative power when it has high resultant magnitude in positive and negative instances. The classifiers in positive and negative bags are independent in the feature space. We represent discriminative classifiers in positive bag by $h_m(\mathbf{x}_p^+)$ (function h is defined in Eq.(1a), where $\mathbf{x}$ represents an instance) and the discriminative classifiers in negative bag as $h_m(\mathbf{x}_n^-)$ where, m stands for any classifier (out of M classifiers) in the classifier pool, and p and n stand for number of instances in the positive bag and the negative bag, respectively. Classifiers from positive and negative instances form a feature space (of dimension equal to the number of positive and negative instances). Classifier from a positive instance in the feature space is independent (orthogonal) of classifier (constructed using same Haar-like feature) from a negative instance. Also, all classifiers from different instances (positive as well as negative) are independent of each other. The resultant classifier score (CSR) for m_{th} classifier can be given by

$$h_m^R = \sqrt{\left(\sum_{p=1}^{N} h_m(\mathbf{x}_p^+)\right)^2 + \left(\sum_{n=1}^{L} h_m(\mathbf{x}_n^-)\right)^2}, \quad \text{for } m = 1:M \tag{6}$$

One problem in Eq.(6) is that if any classifier in any instance (in either bag) differs significantly from the others (does not fit), the resultant will be deviated from the most correct value (i.e., the value of classifiers from actual or tracked target). A solution to this problem is to average the values of classifiers for all the instances. In [40] and [41], it is demonstrated that when there is large errors in the estimations of classifiers, mean value can be used to lower the classification error. This is a type of mean filtering wherein unwanted value (a classifier with a value significantly different from others) can be suppressed. In negative bag, instances are taken around the target (at some distance away) to consider the background. Therefore, classifiers of different negative instances differ by large value and can be averaged for better accuracy. However in positive bag, all the instances are cropped closely and therefore, it is be better to have classifier of the most correct instance as averaging will corrupt the value. In our approach, unlike the MIL, bag likelihood is not calculated by probability model and so the argument of having most correct positive instance holds true. In the negative instances, we replace the classifiers (in same feature space) values with the average (mean) classifier value. In the positive bag, instead of average, we keep the classifier of tracked instance. Hence, the multidimensional space is reduced to 2-dimensional space and Eq.(6) can be written as,

$$h_m^R = \sqrt{\left(h_m(\mathbf{x}_c^+)\right)^2 + \left(h_m^{mean}(\mathbf{x}^-)\right)^2} \tag{7}$$

where $h_m(\mathbf{x}_c^+)\,(m = 1:M)$ are classifiers from most correct positive instance and $h_m^{mean}(\mathbf{x}^-)\,(m = 1:M)$ are the mean

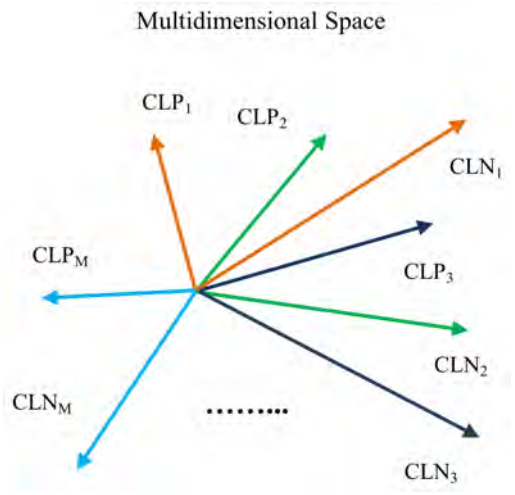


Fig. 1. M classifiers in 2 dimensional space

classifiers from negative instances. Eq.(7) cannot produce better resultant classifier as there is no proper scaling of the two terms inside the square root. If a classifier of a bag is high in magnitude, it will dominate the resultant classification score. Aksoy and Haralick, in [42], discussed the feature normalization independently in the [0, 1] range for improved discrimination power of similarity measures in image database retrieval system. Let $h_m^{mean}(\mathbf{x}^+)$ is the mean taken along all instances in positive bag. If maximum value of $h_m^{mean}(\mathbf{x}^+)$ (M classifiers) is M_p and minimum value of $h_m^{mean}(\mathbf{x}^-)$ is M_n then Eq.(7) is modified as,

$$h_m^R = \sqrt{\left(\frac{w_p \times h_m(\mathbf{x}_c^+)}{M_p}\right)^2 + \left(\frac{w_n \times h_m^{mean}(\mathbf{x}^-)}{M_n}\right)^2} \quad (8)$$

Here, ($w_p = N + L$) and ($w_n = N + \lambda L$) function as weights which make sure that the positive instance classifier is given more weight than the negative instance classifier. The motivation behind N and L for weight is that it will be adjusted according to the number of positive and negative instances as the resultant classifier Eq.(8) consists of mean value of the classifiers in all negative instances as well as the maximum and minimum value of the mean for classifier in positive and negative instances. The value of λ is set to 0.7. The reason for taking minimum value for negative instances is that for instances in negative bag, the classifier log-likelihood ratio according to (1a) becomes negative. The maximization of CSR is equivalent to maximization of its square. The latter will save the computation. Therefore, the classifiers with high discriminative power can be chosen by maximizing squared CSR (SCSR) given by Eq.(9). Our proposed algorithm chooses features which have high SCSR according to (9). The SCSR values are sorted and top K features $\{fid_i\}_{i=1}^K$ are selected from M features pool as given in (10), where function $F_{sort}(\cdot)$ returns the indices that sorts an array.

$$\begin{aligned} \text{SCSR}_m &= \left(h_m^R\right)^2 \\ &= \left(\frac{w_p \times h_m(\mathbf{x}_c^+)}{M_p}\right)^2 + \left(\frac{w_n \times h_m^{mean}(\mathbf{x}^-)}{M_n}\right)^2, \\ &\qquad\qquad \text{for } m = 1 : M \end{aligned} \quad (9)$$

$$\left.\begin{aligned} \mathbf{id} &= F_{sort}\left(-SCSR\right) \\ fid_i &= id_j \text{ , for } i = 1 : K, j = 1 : K \end{aligned}\right\} \quad (10)$$

where, **id** is 1D array containing M elements, $\{id_i\}_{i=1}^M$. Fig.1 shows the representation of M classifiers. The CLP is for positive instance classifier (from most correct positive instance) and the CLN is for negative instance classifier (representing mean of classifiers from all negative instances). The proposed online feature selection procedure is given in Algorithm 1. We believe that the Algorithm 1 can be used as the boosting algorithm for feature or classifier selection in other applications as it selects classifiers which have high discriminating power.

For feature extraction in our MIL based tracking algorithm, we use the Haar-like features f_k, as in [8], [13] and [14] which are normally distributed with four parameters $(\mu_1, \sigma_1, \mu_0, \sigma_0)$. The update of parameters is same as in [13] and [14] and is given by

$$\mu_1 \leftarrow \eta\mu_1 + (1-\eta)\bar{\mu} \quad (11)$$

$$\sigma_1 \leftarrow \sqrt{\eta(\sigma_1)^2 + (1-\eta)(\bar{\sigma})^2 + \eta(1-\eta)(\mu_1 - \bar{\mu})^2} \quad (12)$$

where

$$\bar{\mu} = (1/N)\sum_{j=0|y_i=1}^{N-1} f_k(x_{ij}), \quad (13)$$

and

$$\bar{\sigma} = \sqrt{(1/N)\sum_{j=0|y_i=1}^{N-1} \left(f_k(x_{ij}) - \bar{\mu}\right)^2} \quad (14)$$

The subscript 1 in Eqs.(11) and (12) represents positive samples, N is the number of positive samples and η is the learning rate. Eqs.(11)–(14) are used for L negative samples as well, with parameters (μ_0, σ_0).

Algorithm 1 Online feature selection by squared classifier score (SCSR) maximization

Input: Dataset X^+, X^-, where $X^+ = \{x_{1j}, y_1 = 1, j = 0, ..., N-1\}$, and $X^- = \{x_{0j}, y_0 = 0, j = N, ..., N+L-1\}$, M classifiers

1: Update all M weak classifiers with data X^+, X^-
2: Select the most correct positive example, $\mathbf{x}_c^+$
3: Compute the mean value of m_{th} $(m = 1, .., M)$ classifier from positive instances, $h_m^{mean}(\mathbf{x}^+)$
4: Compute the mean value of m_{th} $(m = 1, .., M)$ classifier from negative instances, $h_m^{mean}(\mathbf{x}^-)$
5: $M_p = \text{MAX}(h_m^{mean}(\mathbf{x}^+))$
6: $M_n = \text{MIN}(h_m^{mean}(\mathbf{x}^-))$
7: Compute weights ($w_p = N + L$) and ($w_n = N + \lambda L$)
8: **for** $m = 1$ to M **do**
9: $\quad \text{SCSR}_m = -((w_p \times h_m(\mathbf{x}_c^+)/M_p)^2 + (w_n \times h_m^{mean}(\mathbf{x}^-)/M_n)^2)$
10: **end for**
11: Feature indices, **id** $= F_{sort}$(SCSR)
12: $h_{k|k=1:K}(\mathbf{x}) = h_{j|j=id_{1:K}}(\mathbf{x})$

Output: Strong classifier, $H(\mathbf{x}) = \sum_{k=1}^{K} h_k(\mathbf{x})$

Our feature selection algorithm based on maximizing the classifier functions values is different from MIL [8], WMIL [13] and ODFS [14] in following ways.

(a) Hardware implementable: In the proposed algorithm, we do not use sigmoid function to model the instance probability as used in all three algorithms (MIL, WMIL and ODFS). This makes the process faster as well as easily implementable in dedicated hardware, e.g., field programmable gate array (FPGA) and application specific integrated circuits (ASICs), for tracking in high frame rate videos.

(b) Reduced time complexity: All three algorithms mentioned above use sorting steps (to sort an array of size M) equal to the required number of features (K) to be selected. The sorting algorithm increases the time complexity. Only one sorting step has been used in our algorithm.

(c) Independence of classifier: We treat each discriminative classifier in positive instance as independent with respect to classifier in negative instance and select the features with high resultant (of positive and negative examples) discriminative power. In MIL, WMIL and ODFS, weak classifiers are selected using greedy method.

(d) Reduced drift problem: Because tracked result in successive frames are not accurate, taking positive samples around it gives noisy data. This problem has been taken into consideration in ODFS by taking the most correct positive instance (tracked result). In our algorithm, the strategy to reduce the noisy positive samples has been taken into consideration. We use most correct positive sample to compute the resultant classifier score.

Fig.2 shows classifier scores from a positive instance in the second frame of Dudek video sequences. The selected feature indices are those where the blue + signs are completely inscribed in red circles. In WMIL, many features are selected where the classifier scores are positive and other features which have high negative scores (higher in magnitude) than those selected are left. In ODFS, quite opposite of WMIL happens. Feature are selected which have higher negative scores while many features of high positive scores are left. In our proposed feature selection algorithm (CSR), features which have high resultant scores (in magnitude) from positive and negative (mean) are selected. The negative mean classifier is not shown here. This analysis is observed in the second frame of Dudek video to show the comparison between WMIL, ODFS and our CSR.

Fig.3 shows top 20 feature rectangles selected by WMIL and Fig.4 shows top 20 rectangles by ODFS. Feature rectangles by our CSR is given in Fig.5. It should be made clear that WMIL and ODFS both use 3 to 4 feature rectangles which are discrete (not side by side as shown here) and compute a feature as the sum of pixels in the rectangles. We have used only 2 feature rectangles both side by side and computed a feature as the difference between sum of pixels in the two rectangles. For comparative analysis, here we use the same 2 feature rectangles in both WMIL and ODFS. This is only to show the difference in feature selection. However for tracking, we use the feature rectangles in WMIL and ODFS as given in their codes. The differences in selected features can be observed more clearly from the classifier scores shown in Fig.2.

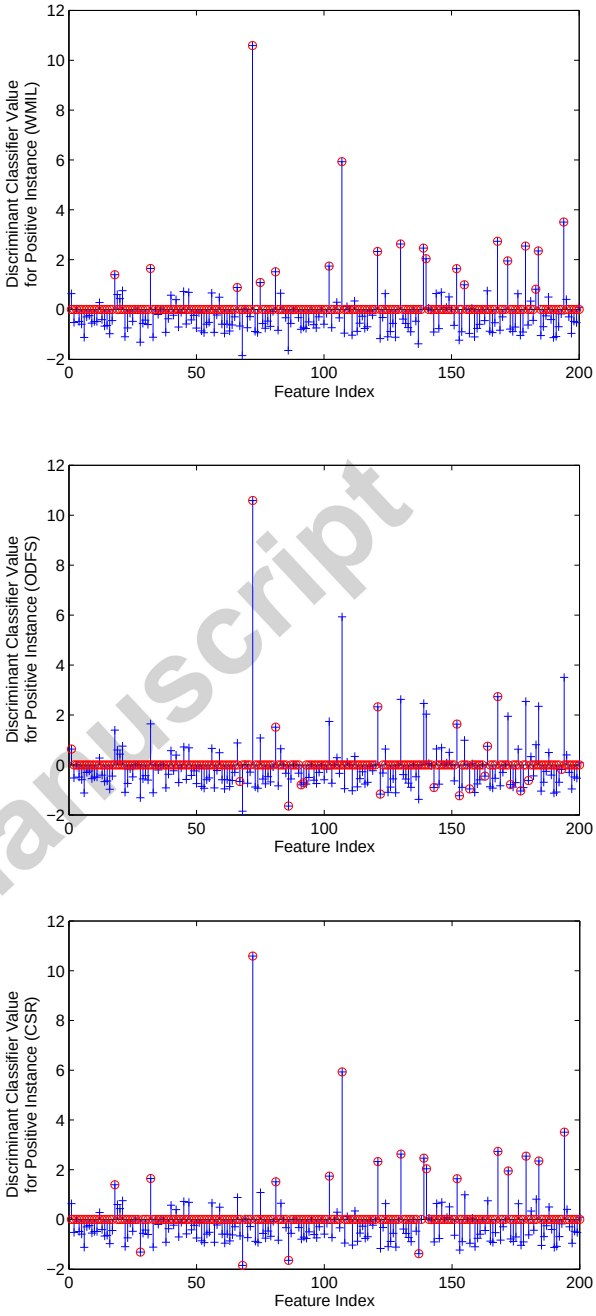
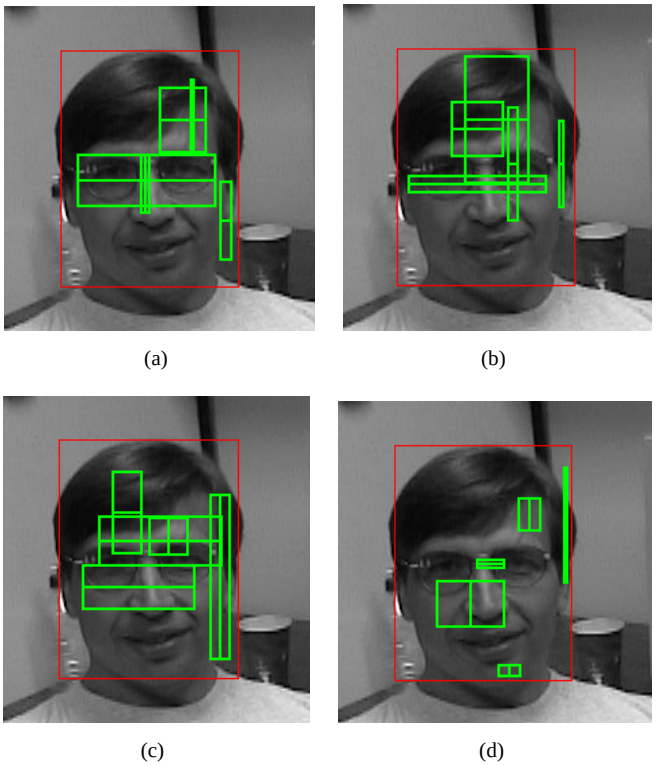


Fig. 2. Comparative selection of features according to classifier values (scores) in a most correct positive instance for second frame of Dudek video in (from top to bottom) WMIL, ODFS, and CSR

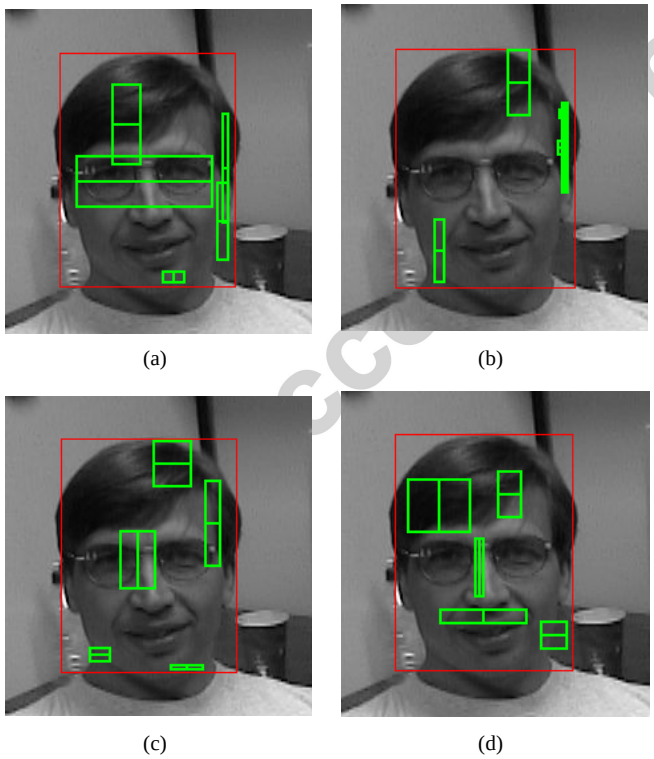
V. Target Tracking Using Classifiers and Haar-Features

Let $\mathbf{x}_{t-1}^T$ be the tracked instance in $(t-1)_{th}$ frame. K highly discriminative features, $\{f_k(\mathbf{x}_{t-1}^T)\}_{k=1}^K$, are selected using the feature selection Algorithm 1. When t_{th} frame arrives, K features are extracted from N_C target candidates $(\{\mathbf{x}_t^s\}_{s=1}^{N_C})$. From each of the features, discriminative classifiers $\{H_K(\mathbf{x}_t^s)\}_{s=1}^{N_C}$ are constructed for all the candidates using Eqs.(1a) and (1b).



(a) (b) (c) (d)

Fig. 3. Feature rectangles (a) 1 to 5, (b) 6 to 10, (c) 11 to 15, and (d) 16 to 20 selected by WMIL in second frame of Dudek video



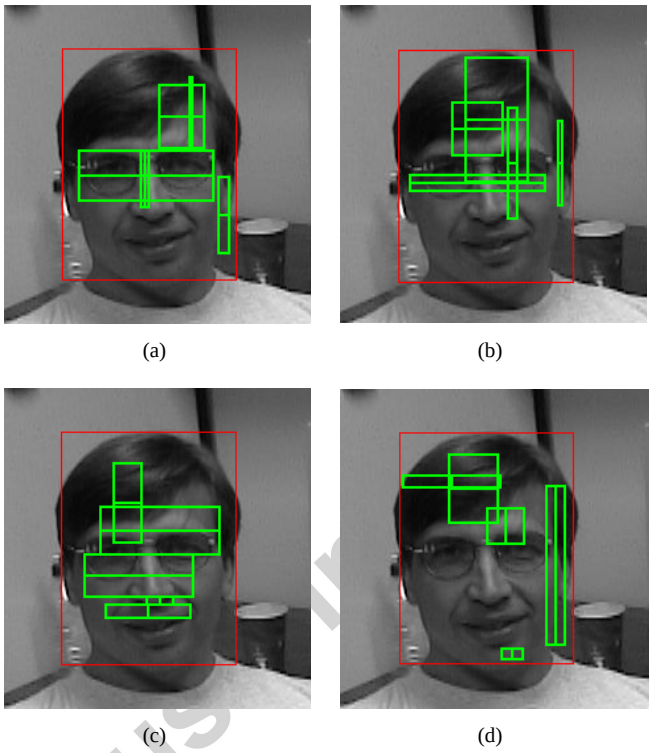
(a) (b) (c) (d)

Fig. 4. Feature rectangles (a) 1 to 5, (b) 6 to 10, (c) 11 to 15, and (d) 16 to 20 selected by ODFS in second frame of Dudek video

A. Intermediate Tracking Instance Using Classifiers

The tracked instance $\mathbf{x}_t^T$ in the current frame t_{th} is given as,

$$\mathbf{x}_t^T = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} H_K(\mathbf{x}_t^s) \tag{15}$$



(a) (b) (c) (d)

Fig. 5. Feature rectangles (a) 1 to 5, (b) 6 to 10, (c) 11 to 15, and (d) 16 to 20 selected by CSR in second frame of Dudek video

In our proposed method, the tracked instance is given by

$$\mathbf{x}_t^T = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} \left(\frac{H_{Ks}(\mathbf{x}_t^s)}{Ks} + \frac{H_K(\mathbf{x}_t^s)}{K} \right) \tag{16}$$

where, Ks is subset of K and Ks represents features having higher SCSR. For this work, Ks is 10 (i.e., Ks contains top 10 features with highest SCSR among K).

B. Kernel Trick on Haar-like Features

Feature similarity can be used to select an instance out of N_C. Hou et al. in [43] used inner product as the metric to measure the similarity between two feature templates. The speciality of finding the similarity measure using inner product is that we get the similarity in very high dimension space using a computationally efficient method, called kernel trick.

We use feature matching in high dimension space using kernel trick to select the most similar (or tracked) instance $\mathbf{x}_t^T$ in the current frame t_{th} as,

$$\mathbf{x}_t^T = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} \mathcal{K}(\widehat{f_K}(\mathbf{x}_{t-1}^T), \widehat{f_K}(\mathbf{x}_t^s)) \tag{17}$$

where, $\mathcal{K}$ is Mercer kernel and $\widehat{f_K}$ represents the feature vector of dimension K and unit magnitude. The difference between Eq.(15) and Eq.(17), apart from kernel trick, is that Eq.(17) is independent of classifiers whose parameters are updated according to Eqs.(11), and (12) using a learning rate.

C. Intermediate Tracking Instance Using Classifiers

The update method of classifier, sometimes, may not be accurate due to occlusion or some other variations. However,

Algorithm 2 Target Tracking using Discriminative Classifiers, Features and Kernel with scale adaptation (CSRKS)

Input: Frame, t, Number of features K and Ks, Initial target patch $\mathbf{x}_{target}$, Learning rate R_1

if $t = 1$ **then**
  $\mathbf{x}_1^T = \mathbf{x}_{target}$
  Select K features f_K^1, $(k = 1 : K)$ using Algorithm 1
  Select a subset Ks, Kth, Kbh, Klh and Krh from K
else
  for $t = 2$ to t **do**
    Extract K features $\{f_K^{t-1}(\mathbf{x}_t^s)\}_{s=1}^{N_C}$ from N_C target candidates ($\{\mathbf{x}_t^s\}_{s=1}^{N_C}$)
    Form discriminative classifiers $H_K(\mathbf{x}_t^s)$, $H_{Kth}(\mathbf{x}_t^s)$, $H_{Kbh}(\mathbf{x}_t^s)$, $H_{Klh}(\mathbf{x}_t^s)$, $H_{Krh}(\mathbf{x}_t^s)$ for $s = 1 : N_C$ candidates using Eqs.(1a), (1b) and features f^{t-1}
    Find the intermediate tracking instance $\mathbf{x}_t^{T_1} = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} H_K(\mathbf{x}_t^s)$
    Find the intermediate tracking instance $\mathbf{x}_t^{T_2} = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} H_{Ks}(\mathbf{x}_t^s)$
    Find the intermediate tracking instance $\mathbf{x}_t^{T_3} = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} \mathcal{K}(\widehat{f_K^{t-1}}(\mathbf{x}_{t-1}^T), \widehat{f_K^{t-1}}(\mathbf{x}_t^s))$ for feature dimension K
    Find the intermediate tracking instance $\mathbf{x}_t^{T_4} = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} \mathcal{K}(\widehat{f_K^{t-1}}(\mathbf{x}_{t-1}^T), \widehat{f_K^{t-1}}(\mathbf{x}_t^s))$ for feature dimension K_S
    Find the intermediate tracking instance $\mathbf{x}_t^{T_5} = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} H_{Kth}(\mathbf{x}_t^s)$
    Find the intermediate tracking instance $\mathbf{x}_t^{T_6} = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} H_{Kbh}(\mathbf{x}_t^s)$
    Find the intermediate tracking instance $\mathbf{x}_t^{T_7} = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} H_{Klh}(\mathbf{x}_t^s)$
    Find the intermediate tracking instance $\mathbf{x}_t^{T_8} = \underset{\mathbf{x}_t^s}{\operatorname{argmax}} H_{Krh}(\mathbf{x}_t^s)$
    Final estimated (tracked) target $\mathbf{x}_t^T = \underset{\mathbf{x}_t^{T_i}}{\operatorname{argmax}} F_c(\mathbf{x}_{target}, \mathbf{x}_t^{T_i})$
    Select K features f_K^t, $(k = 1 : K)$ using Algorithm 1
    Select a subset Ks, Kth, Kbh, Klh and Krh from K
    $\mathbf{x}_{target} = (1 - R_1) \times \mathbf{x}_{target} + R_1 \times \mathbf{x}_t^T$
  end for
end if

Output: Tracked location of $\mathbf{x}_t^T$

using only features for the classification method in Eq.(17) may not perform better all the time. We used the following tracking strategy shown in Algorithm 2 to make sure that the most correct candidate being selected is as similar as possible to the target. We name the algorithm CSRKS (CSR with kernel and scale adaptation).

In CSRKS, we obtain 4 (for target size smaller than a threshold) and 8 (for target size greater than a threshold) intermediate tracked instances in parallel from N_C target candidates.

For getting 4 intermediate instances (for target size smaller than a threshold): For getting first two, we use classifiers while for other two, we use feature matching. Using classifiers, we crop two tracked instances out of N_C by Eq.(15). Using Eq. (15), the first intermediate tracked instance $\mathbf{x}_t^{T_1}$ is obtained using full K classifiers while the second one $\mathbf{x}_t^{T_2}$ is obtained using a subset Ks of K. The Ks is the set of features that have higher SCSR from Algorithm 1. In the similar way, we use feature similarity given by Eq.(17) for cropping two more intermediate instances $\mathbf{x}_t^{T_3}$ and $\mathbf{x}_t^{T_4}$, corresponding to feature sets K and Ks.

For getting 4 additional intermediate instances (for target size greater than a threshold): Two intermediate instances ($\mathbf{x}_t^{T_5}$ and $\mathbf{x}_t^{T_6}$) are obtain from classifiers (Eq. (15)) whose Haar-like features are either in the top half (Kth) or in the bottom half (Kbh) of the target region. Similarly, two more instances ($\mathbf{x}_t^{T_7}$ and $\mathbf{x}_t^{T_8}$) are obtained using classifiers whose Haar-like features are in the left half (Klh) or in the right half (Krh) of the target space. This strategy is applied to assimilate the part based tracking feature. For smaller target size, very few features will be available in half part of the target space. Therefore, the part based strategy may not be effective. The target size is computed as, target size = $\sqrt{h \times w}$, where, h and w are, respectively, the height and the width of the target. For this work, threshold (target size) for cropping the 4 additional instances is taken as 45. Fig. 6 shows the selected feature rectangles in complete as well as half target space for David video sequence.

The cropped instances are compared for their 1D correlations with the updated target template $(\mathbf{x}_{target})$. To adapt the scale variation of the target, around center of each intermediate tracked instance, 4 additional instances are cropped with their scaled sizes. The 2 instances are obtained using up scaling (height and width is increased at fixed target size) while another 2 instances are obtained using down scaling (height and width is decreased at fixed target size). In this way, there

are $5 \times 8 = 40$ instances for comparing 1D correlations. Histogram of oriented gradient (HOG) features [44] have been used for comparing 1D correlations. The HOG features are to some extent invariant to local geometric and photometric transformations [45].

In Algorithm 2, $F_c(\mathbf{x}_i, \mathbf{x}_j)$ represents the function for computing 1D correlation between $\mathbf{x}_i$ and $\mathbf{x}_j$. The final tracked instance $\mathbf{x}_t^T$ is the one which has the highest correlation among them. The purpose of using this strategy is to avoid the suboptimal tracked instance by using different kinds of feature and classifier sets. The target template is updated using a learning rate R_1, $(0 < R_1 < 1)$.

D. Background Features for Smaller Target Size

For smaller target, occlusion and deformations severely affect the detection task. It is better to exploit the background around the target to mitigate this problem [33]. Therefore, we run a parallel CSR algorithm with higher target area (width and height increased to 1.5 times the original) for smaller target size (less than 60 in this work). Using this algorithm, an intermediate tracking instance $(\mathbf{x}_t^{T_{ss}})$ with size equal to the original target is cropped and its 1D correlation ($corr_S$) with the updated target is computed. The 1D correlation ($corr_O$) is also computed between $\mathbf{x}_{target}$ and $\mathbf{x}_t^T$. At the end of the main tracking algorithm (2), two correlation ($corr_S$) and ($corr_O$) are compared to get the final tracked instance. However, $corr_S$ is given lesser weight than $corr_O$ (0.92 against 1 in this work). This is to have more confidence in the tracking instance obtained using CSRKS Algorithm 2.

VI. Experimental Setups and Results

We extract only 2 rectangular (horizontal and vertical) Haar-like features as opposed to 3 to 4 rectangular features in WMIL and ODFS. For our CSR, we use feature pool of size $M = 250$. Using classifier score maximization algorithm (Algorithm 1), we select $K = 20$ features for target size less than 30, $K = 25$ features for target size between 30 and 120, and $K = 30$ features for target size greater than 120. The algorithms WMIL and ODFS used $M = 150$ features and selected $K = 15$ features. For our CSRKS algorithm (the feature pool is M), we select same number of features as that of CSR for target size less than 45 (where 4 additional intermediate instances for features in half target spaces are not used). However, to have the enough features in half target space, the number of selected features is increased to 30 for target size between 45 and 120. Also, for target size more than 120, we use $M = 300$ and $K = 40$.

We vary the radius, α, for cropping the positive samples and the search radius for finding the target according to the target size. It is given in Table I. Other parameters required for CSRKS is given in Table II. The negative samples are cropped between radii, $(\gamma = 2\alpha)$ and $(\beta = 1.5\zeta)$ where ζ is search radius for finding a new object location in successive frames. The reason for setting varied radii is that for smaller target size, smaller α will allow dense negative samples around the target region. For WMIL and ODFS, radii for cropping negative samples (between $(\gamma = 2\alpha)$ and $(\beta = 1.5\zeta)$) as given

TABLE I
Search Window and Radius Settings for CSR and CSRKS

Mean target size	≤ 30	≤ 45	≤ 60	All others
ζ	20	25	30	30
α	1	2.5	2.5	3.5

TABLE II
Parameters for CSRKS

K_S	R_1	σ
10	0.012	0.1

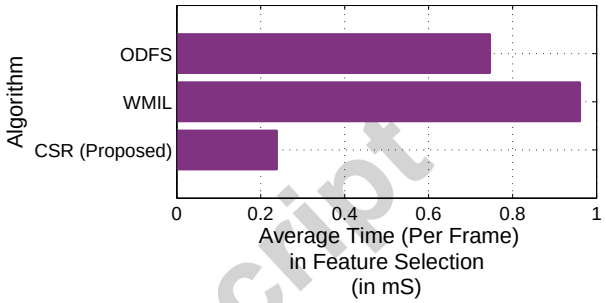


Fig. 7. Average feature selection time in WMIL, ODFS and CSR

in the respective papers are used. Learning rate parameter used in the proposed algorithms is $\eta = 0.9$). For running parallel CSR algorithm for smaller target size (as mentioned in sub section V-D), we choose $M = 300$, $\zeta = 25$, $K = 30$ for target size less than 45 and $K = 40$ for other target size.

We use the best learning rates $(\eta = 0.85)$ for WMIL and $(\eta = 0.93)$ for ODFS as mentioned in respective papers. The upper limit on the number of negative samples in our algorithms is set to 80. In KCF, STRUCK and TLD, all parameter are set as given in respective codes for best performance. We use a budget size 100 for better performance in STRUCK. With one parameter, a trackers may perform better in some videos while it may not perform better in some other. However, we maintain the best and unique parameters in all the trackers for a fair evaluation across the diverse set of videos.

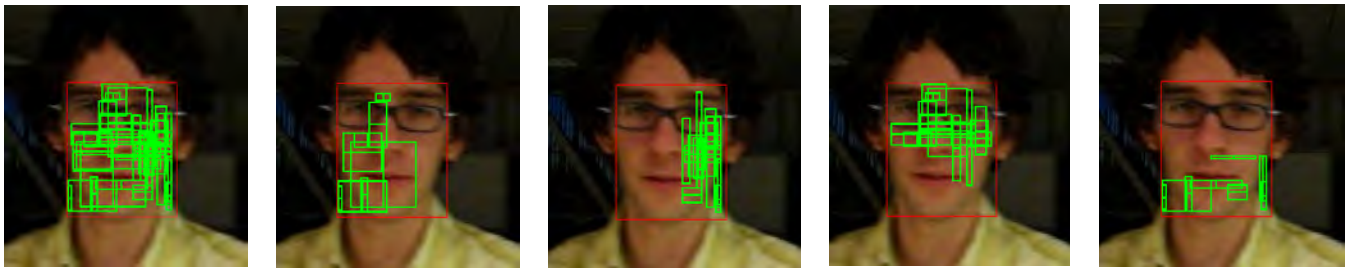
A simplified form of weak classifier given in Eq.1(a) can be written as $h_k = \ln(\sigma_0/\sigma_1) + \left((f_k(\mathbf{x}) - \mu_0)^2/2{\sigma_0}^2\right) - \left((f_k(\mathbf{x}) - \mu_1)^2/2{\sigma_1}^2\right)$. We use this form, a faster one, in our implementation. It should be made clear that it is a simplified form only for implementation, but there is no functional difference with the form used in WMIL and ODFS. For our proposed CSRKS algorithm, we use the Gaussian kernel given by

$$\mathcal{K}(f(\mathbf{x}_1), f(\mathbf{x}_2)) = \exp\left(\frac{-\|f(\mathbf{x}_1) - f(\mathbf{x}_2)\|^2}{2\sigma^2}\right) \text{ for } \sigma > 0 \tag{18}$$

The value of parameter σ is given in Table II.

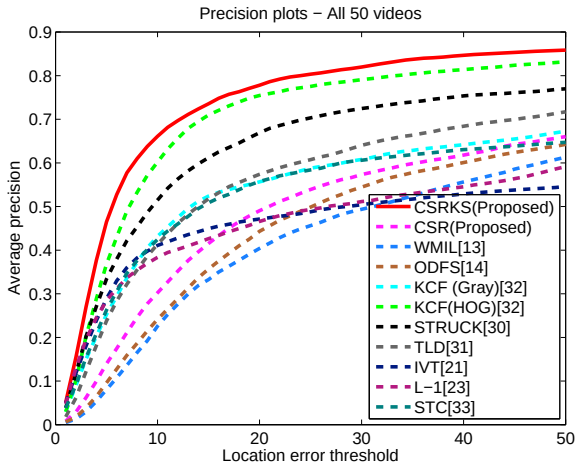
A. Simulation Environments and Datasets

We use the Intel core $i7$ Processor (3.4 GHz frequency) on Windows XP platform with 2 GB of RAM to simulate all the reference codes and codes of the proposed algorithms. The

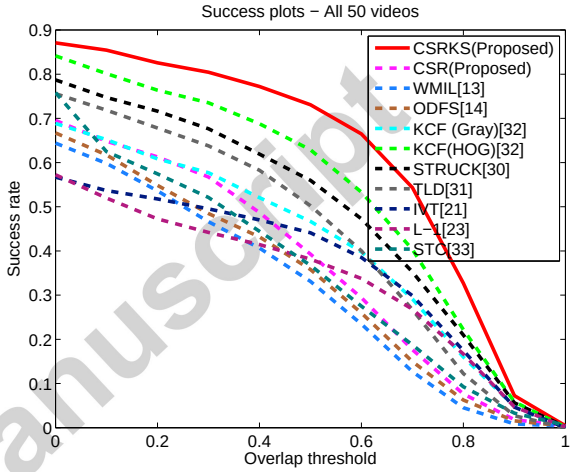


(a) (b) (c) (d) (e)

Fig. 6. Third frame of David sequence containing selected feature rectangles for target space (a) full, (b) left half, (c) right half, (d) top half, and (e) bottom half. The half feature rectangles in left half and top half can cross the half target space to cover the target center

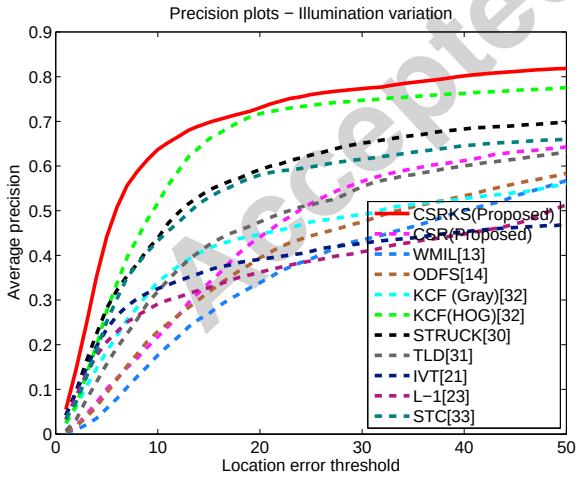


(a) Precision for all 50 videos

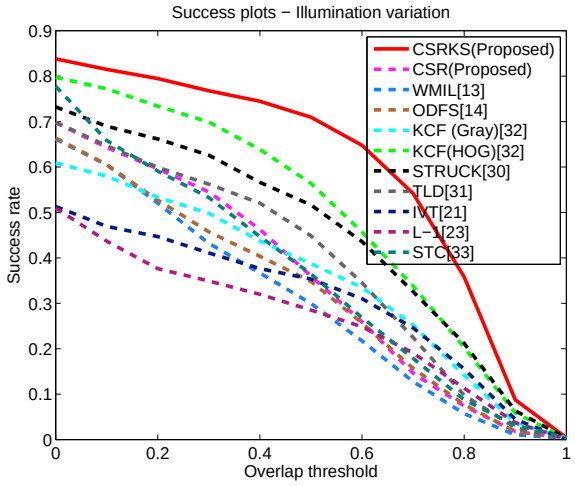


(b) Success for all 50 videos

Fig. 8. Precision and success plots for all 50 videos



(a) Precision for illumination variation



(b) Success for illumination variation

Fig. 9. Precision and success plots illumination variation

MATLAB version used is R2012a. We compile the STRUCK code in MS Visual Studio 2010, with OpenCV Version 2.2. The simulations are performed on total 50 video sequences. The datasets containing 50 challenging video sequences are taken from CVPR 2013 paper available on Web[1][46]. These video sequences contain both static and changing backgrounds, high clutter, partial occlusion, changed appearances, and motion blur. For all the videos, the tracking results are obtained

[1]https://sites.google.com/site/wuyi2018/benchmark

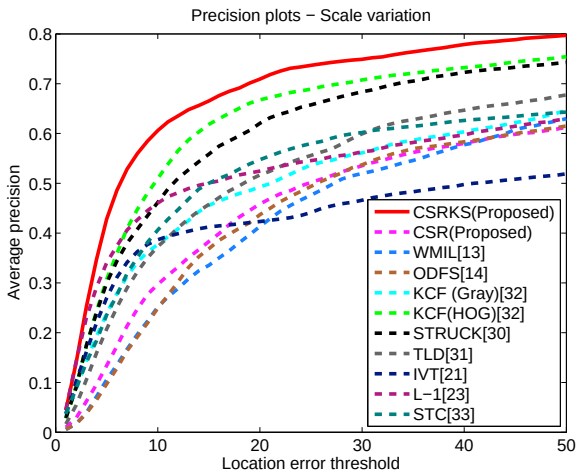
TABLE III
AVERAGE CENTER ERROR (CE) AND AVERAGE OVERLAP (OV) SCORE OF DIFFERENT TRACKERS

		WMIL [13]	ODFS [14]	CSR (Proposed)	KCF [32]	KCF (HOG) [32]	STRUCK [30]	TLD [31]	IVT [21]	L-1 [23]	STC [33]	CSRKS (Proposed)
Basketball	CE	260.5	12.5	12.8	126.9	**7.8**	171.8	279.2	11.8	58.0	75.2	*5.9*
	OV	0.021	0.611	0.615	0.227	**0.676**	0.026	0.0218	0.440	0.072	0.287	*0.732*
Boy	CE	8.9	79.2	6.0	3.0	*2.8*	3.8	4.0	10.2	72.7	25.9	**2.9**
	OV	0.550	0.241	0.667	**0.773**	*0.777*	0.760	0.603	0.678	0.356	0.543	0.769
Car4	CE	116.5	73.7	71.1	36.0	9.8	**8.6**	111.0	136.6	129.7	10.6	*2.7*
	OV	0.148	0.170	0.203	0.354	0.483	**0.489**	0.252	0.233	0.140	0.351	*0.881*
CarDark	CE	101.0	32.2	4.6	4.4	6.0	*0.9*	5.9	1.1	18.0	2.8	**1.0**
	OV	0.032	0.445	0.654	0.694	0.614	*0.892*	0.631	0.826	0.547	0.750	**0.864**
Coke11	CE	38.8	50.9	29.4	18.3	18.6	*12.2*	18.5	103.6	108.0	74.3	**18.0**
	OV	0.272	0.224	0.382	**0.588**	0.549	*0.648*	0.491	0.029	0.055	0.104	0.557
Crossing	CE	19.1	8.6	5.0	8.1	*2.2*	119.8	45.5	57.5	72.6	34.0	**3.3**
	OV	0.439	0.535	0.663	0.566	**0.710**	0.304	0.170	0.329	0.204	0.248	*0.742*
David	CE	28.8	63.9	23.4	14.0	**8.0**	60.2	33.1	9.3	13.7	12.1	*3.3*
	OV	0.362	0.244	0.380	0.470	0.538	0.236	0.443	**0.565**	0.507	0.522	*0.829*
David2	CE	70.3	19.6	5.9	5.3	2.0	*1.4*	5.8	**1.6**	10.3	5.5	1.8
	OV	0.038	0.219	0.636	0.663	0.827	*0.870*	0.633	**0.835**	0.676	0.588	0.807
David3	CE	66.9	12.5	10.7	95.0	*4.3*	106.4	300.2	100.8	143.3	6.3	**5.1**
	OV	0.330	0.687	0.673	0.289	*0.772*	0.292	0.110	0.311	0.257	0.432	**0.755**
Deer	CE	15.55	91.2	7.2	*4.5*	21.1	**5.2**	34.5	133.6	172.0	400.8	5.8
	OV	0.599	0.285	0.706	**0.749**	0.623	0.739	0.488	0.252	0.042	0.040	*0.765*
Doll	CE	16.7	12.0	14.7	36.8	8.3	8.7	7.7	16.5	*3.8*	15.2	**7.2**
	OV	0.492	0.509	0.513	0.413	0.534	0.542	0.623	0.621	*0.662*	0.317	**0.629**
Dudek	CE	34.9	27.4	40.7	15.3	**12.0**	*11.4*	25.1	19.1	110.0	25.5	14.3
	OV	0.579	0.629	0.545	0.710	0.727	**0.730**	0.657	0.679	0.613	0.587	*0.774*
Faceocc	CE	28.4	23.2	49.6	*12.0*	13.9	18.3	40.7	16.6	14.8	250.4	**13.4**
	OV	0.600	0.681	0.476	*0.789*	0.774	0.727	0.522	0.752	0.766	0.186	**0.782**
Faceocc2	CE	43.1	12.5	10.6	**5.5**	7.6	5.9	18.7	14.2	13.4	10.1	*4.8*
	OV	0.333	0.685	0.701	0.781	0.751	**0.785**	0.562	0.672	0.366	0.689	*0.818*
FleetFace	CE	109.1	55.9	33.6	25.0	25.5	23.0	65.9	**22.4**	*21.1*	85.0	27.1
	OV	0.305	0.554	0.539	0.595	0.589	**0.608**	0.507	0.607	0.548	0.419	*0.629*
Football	CE	16.0	15.9	19.5	**6.5**	5.8	86.9	10.6	203.5	18.6	10.0	*3.6*
	OV	0.490	0.521	0.486	0.733	**0.762**	0.332	0.637	0.248	0.491	0.593	*0.804*
Freeman1	CE	15.6	62.1	*11.2*	113.3	94.8	14.2	**13.6**	111.9	96.5	114.9	16.6
	OV	0.281	0.122	**0.341**	0.221	0.214	0.337	0.336	0.156	0.193	0.185	*0.424*
Girl	CE	41.0	20.0	13.9	25.0	11.9	13.0	9.4	24.2	*3.4*	21.8	**7.5**
	OV	0.086	0.255	0.413	0.519	0.545	0.464	**0.576**	0.414	0.524	0.334	*0.599*
Jumping	CE	54.8	99.1	50.4	31.8	26.1	6.5	6.6	**5.5**	57.7	67.2	*3.8*
	OV	0.031	0.012	0.102	0.166	0.274	0.616	**0.667**	0.651	0.109	0.069	*0.748*
Mhyang	CE	43.6	30.6	12.7	3.4	3.9	*2.5*	4.3	3.6	4.1	4.5	**3.3**
	OV	0.226	0.387	0.618	0.795	0.796	**0.817**	0.813	0.794	0.798	0.688	*0.835*
Shaking	CE	11.0	10.2	140.9	228.4	112.5	50.0	34.7	126.7	92.2	**9.6**	*5.9*
	OV	0.650	**0.661**	0.146	0.008	0.039	0.290	0.401	0.009	0.012	0.623	*0.779*
Soccer	CE	84.3	128.2	29.7	212.0	*15.3*	131.5	75.1	174.9	115.8	245.4	**23.4**
	OV	0.140	0.210	0.302	0.197	**0.422**	0.149	0.125	0.171	0.133	0.102	*0.484*
Tiger1	CE	104.5	105.7	27.6	44.9	*8.0*	16.8	21.2	106.7	83.0	63.5	**10.5**
	OV	0.108	0.133	0.476	0.457	*0.785*	0.616	0.601	0.215	0.146	0.261	**0.712**
Tiger2	CE	28.2	53.9	28.1	163.8	47.4	**20.6**	44.7	73.3	72.3	57.4	*16.9*
	OV	0.453	0.160	0.445	0.130	0.354	**0.555**	0.377	0.182	0.112	0.110	*0.595*
Singer1	CE	16.7	18.7	18.3	19.6	10.7	13.6	39.3	23.2	**4.0**	5.7	*3.8*
	OV	0.351	0.330	0.353	0.355	0.355	0.358	0.481	0.377	**0.766**	0.531	*0.823*
Singer2	CE	12.1	56.4	114.6	169.7	**10.2**	174.1	55.1	192.8	191.7	52.7	*8.7*
	OV	0.706	0.474	0.156	0.048	**0.732**	0.041	0.234	0.046	0.033	0.408	*0.758*
Sylvester	CE	18.2	21.5	41.1	13.8	12.9	**6.2**	11.9	76.4	35.4	9.4	*5.1*
	OV	0.546	0.513	0.309	0.632	0.643	*0.723*	0.595	0.279	0.356	0.524	**0.711**
Walking	CE	12.4	10.3	182.1	10.3	3.9	3.8	13.7	5.3	*2.3*	7.1	**3.5**
	OV	0.475	0.497	0.190	0.450	0.530	0.571	0.445	*0.667*	0.596	0.598	**0.646**
Walking2	CE	57.6	65.1	52.1	39.1	28.9	11.7	53.9	72.2	**3.2**	13.8	*2.4*
	OV	0.264	0.220	0.292	0.313	0.395	0.510	0.214	0.334	**0.762**	0.518	*0.813*
Woman	CE	130.3	121.3	121.8	153.5	10.0	*4.1*	139.4	246.9	147.6	26.4	*4.1*
	OV	0.110	0.124	0.135	0.188	0.705	**0.733**	0.126	0.130	0.139	0.353	*0.751*

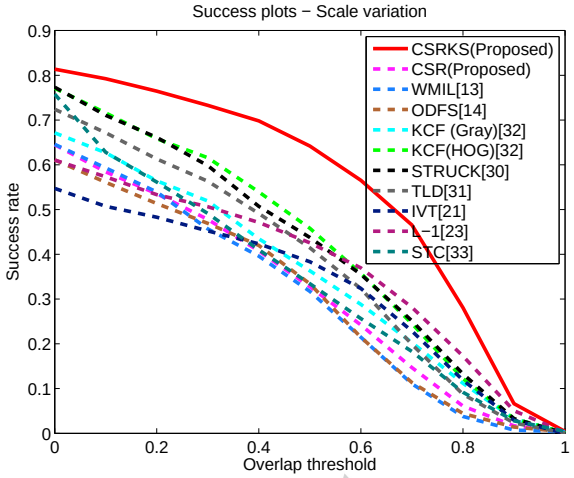
with same set of parameters in the code, i.e., we do not tune the parameter for a specific video for any algorithm used for comparison.

B. Experimental Results

Fig. 7 indicates the average feature selection time in 50 video sequences for WMIL, ODFS and our CSR. For WMIL and ODFS algorithms, reported time is the time measured for selection of 15 features, while for CSR, the measured time

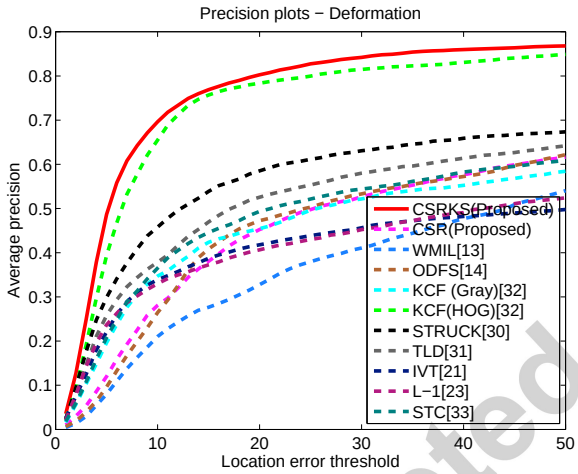


(a) Precision for scale variation

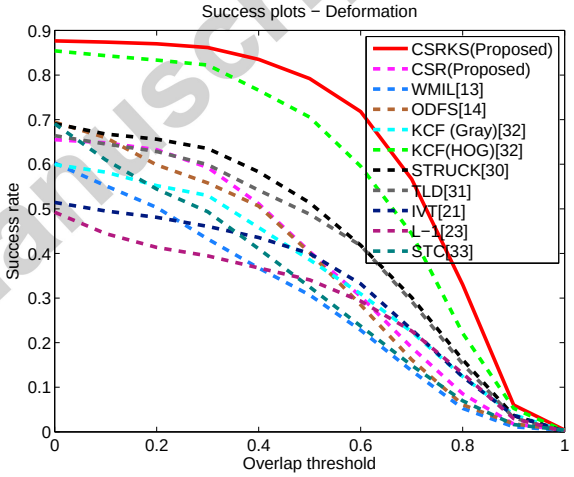


(b) Success for scale variation

Fig. 10. Precision and success plots for scale variation

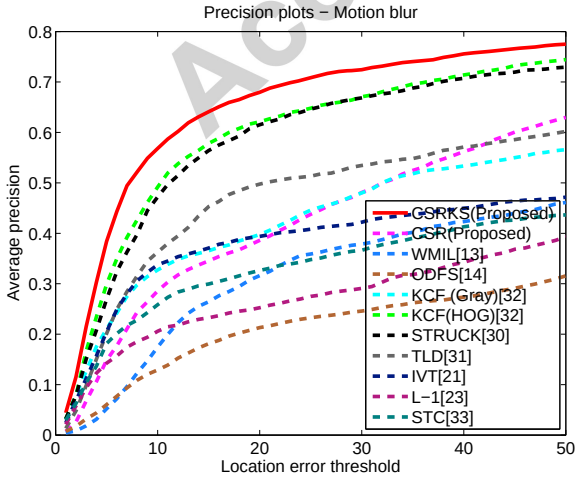


(a) Precision for deformation

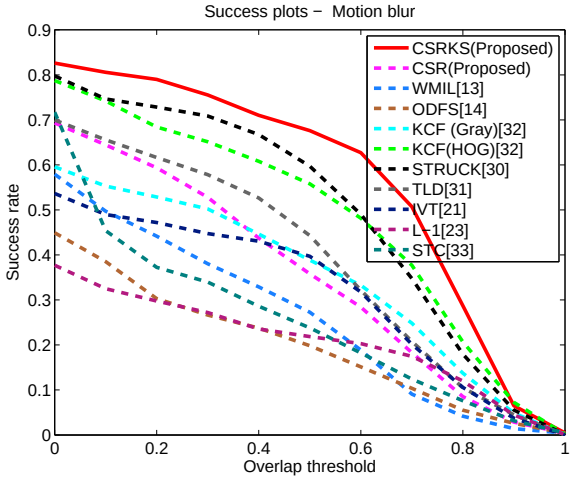


(b) Success for deformation

Fig. 11. Precision and success plots for deformation

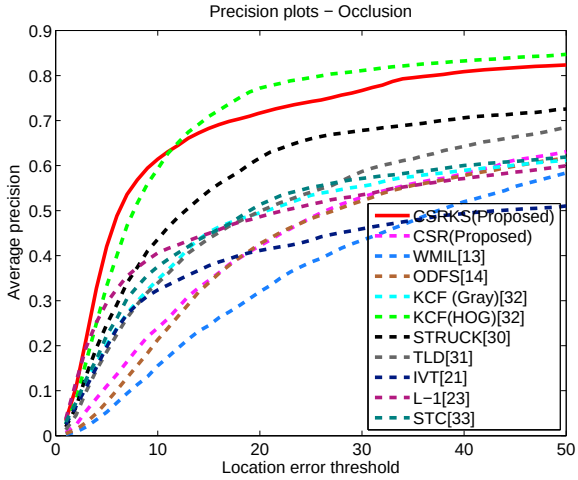


(a) Precision for motion blur

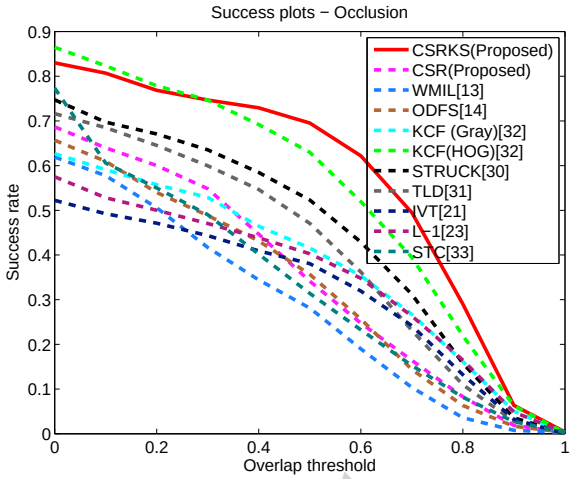


(b) Success for motion blur

Fig. 12. Precision and success plots for motion blur

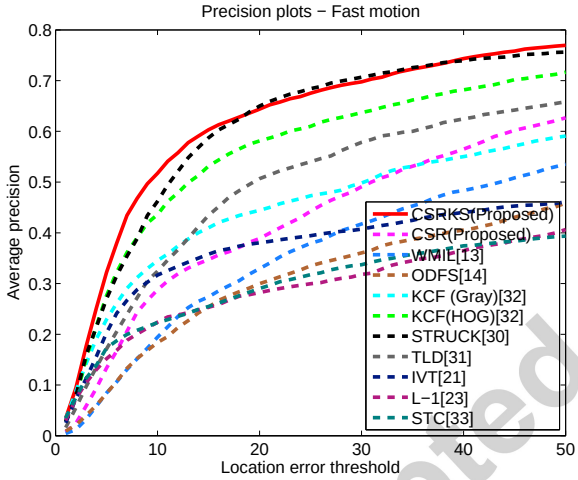


(a) Precision for occlusion

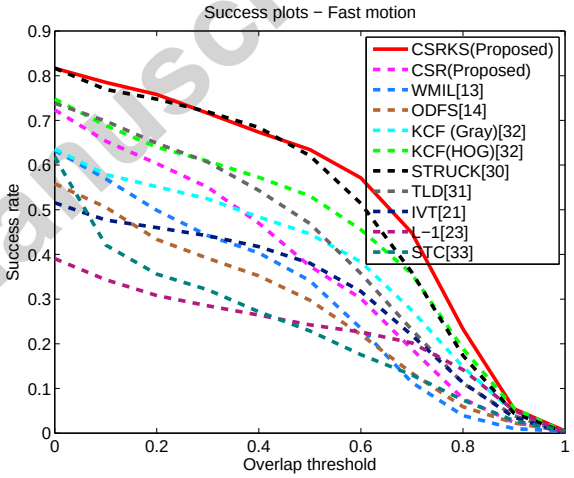


(b) Success for occlusion

Fig. 13. Precision and success plots for occlusion



(a) Precision for fast motion



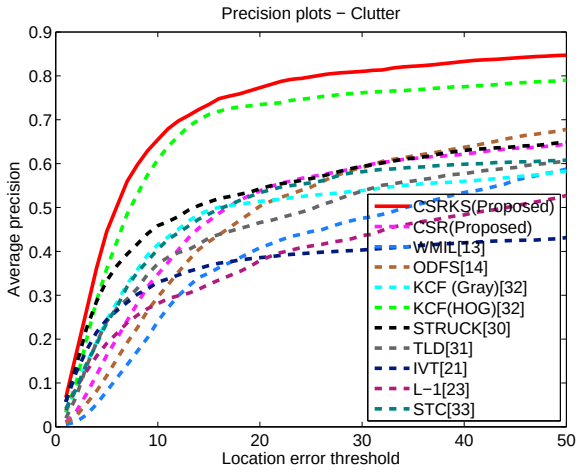
(b) Success for fast motion

Fig. 14. Precision and success plots for fast motion

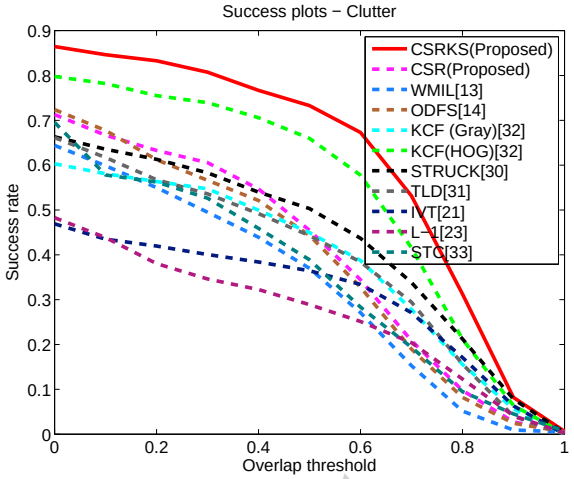
is for the selection of 20 to 30 features. Our CSR algorithm selects feature very quickly as compared to WMIL and ODFS algorithms. The center location error is the Euclidean distance between the ground truth center and the tracked center. Overlap score (OV) is another quantitative measurement parameter which is defined as, $OV = \frac{|B_T \bigcap B_G|}{|B_T \bigcup B_G|}$, where B_T is bounding box of the tracked object, B_G is the bounding box of the ground truth, $\bigcap$ and $\bigcup$ represent the intersection and union respectively, and $|.|$ is for number of pixels in the region [46]. The average center location error (CE) and average overlap score (OV) for some 30 video sequences are shown in Table III. Red font indicates the best accuracy while blue font is used for representing the second best accuracy. In most of the video sequences, the performance of our CSRKS algorithm is best or second best.

Precision and success plots are two quantitative evaluation methods. Precision curve indicates the percentage of frames which have predicted target location within some specified threshold of the ground truth. The success plot shows the percentage of frames where overlap score (OV) is larger than a specified threshold. Fig. 8 to Fig. 17 show the precision and success plots for different tracking algorithms. Our CSRKS algorithm performs best in all challenges except in occlusion. In videos containing target occlusion, KCF(HOG) performs better in precision evaluation as it efficiently utilizes background surrounding the target in tracking by detection method. The tracked rectangles in some of the video sequences are shown in Fig. 18 and Fig. 19.

The average tracking speed of our CSR is approx. 100 frames per second (FPS). WMIL and ODFS have average tracking speed of approx. 92 FPS. Tracking speed of CSRKS is reduced to 13.4 FPS due to HOG feature computation in 40 instances. The STC has the highest average tacking speed of approx. 381 FPS, but it shows the poor tracking results in various sequences. The average tracking speed (in terms of FPS) of L-1, STRUCK, IVT, TLD, KCF and KCF (HOG) are

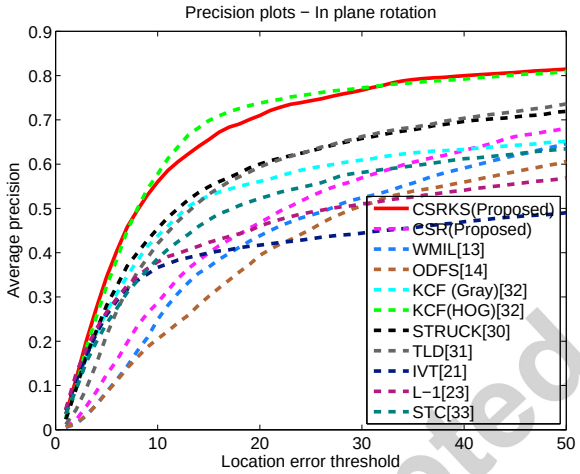


(a) Precision for clutter

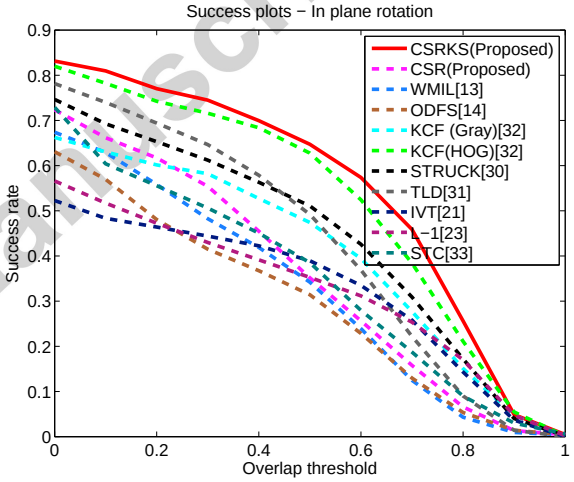


(b) Success for clutter

Fig. 15. Precision and success plots for clutter



(a) Precision for in plane rotation



(b) Success for in plane rotation

Fig. 16. Precision and success plots for in plane rotation

approx. 9.4, 13.9, 17.4, 48.7, 202 and 171.7, respectively.

VII. Conclusion

Fast feature selection algorithm (namely CSR) in MIL framework is performed by normalized sum of classifiers scores of most accurate instance in positive bag and mean of classifiers scores in negative bag. In addition of doing high speed feature selection, its tracking performance is better than the existing feature selection algorithms in that framework. To enhance the tracking performance, we used another algorithm called CSRKS. In CSRKS, different intermediate instances with different scales are cropped using classifiers and kernel based feature similarity. A Gaussian kernel is used to measure the feature similarity. The final estimated instance is the one which has the highest correlation with respect to the updated target. The tracking accuracy of CSRKS is better than the other state-of-the-art trackers.

References

[1] G. Tsagkatakis and A. Savakis, "Online distance metric learning for object tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 12, pp. 1810–1821, 2011.

[2] U. Ali and M. B. Malik, "Hardware/software co-design of a real-time kernel based tracking system," *Journal of Systems Architecture*, vol. 56, no. 8, pp. 317–326, 2010.

[3] F.-S. Chen, C.-M. Fu, and C.-L. Huang, "Hand gesture recognition using a real-time tracking method and hidden markov models," *Image and vision computing*, vol. 21, no. 8, pp. 745–758, 2003.

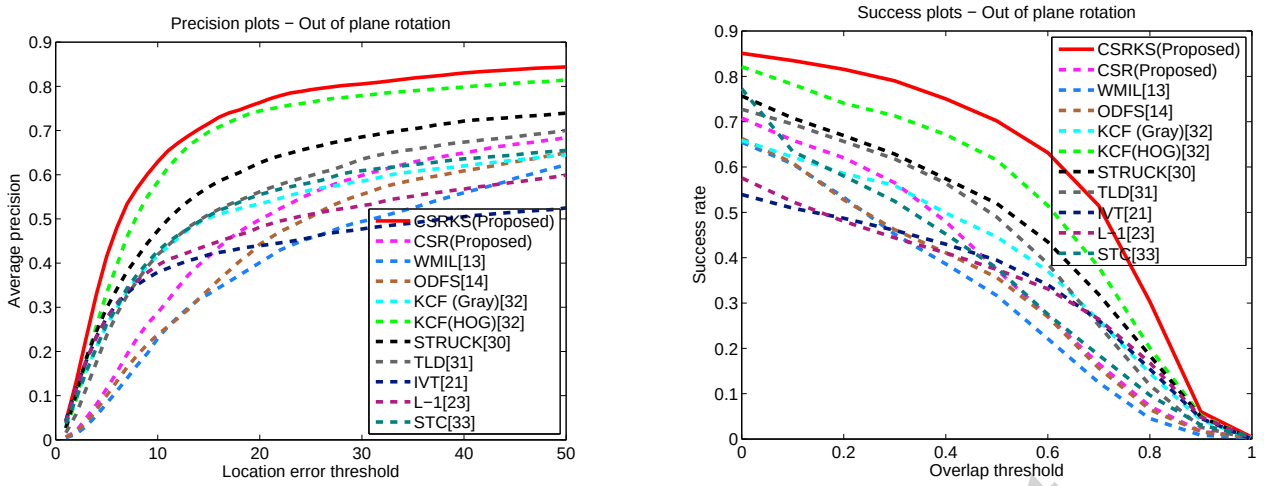
[4] S. Chen, "Kalman filter for robot vision: a survey," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, 2012.

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

[6] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting." in *BMVC*, vol. 1, no. 5, 2006, p. 6.

[7] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

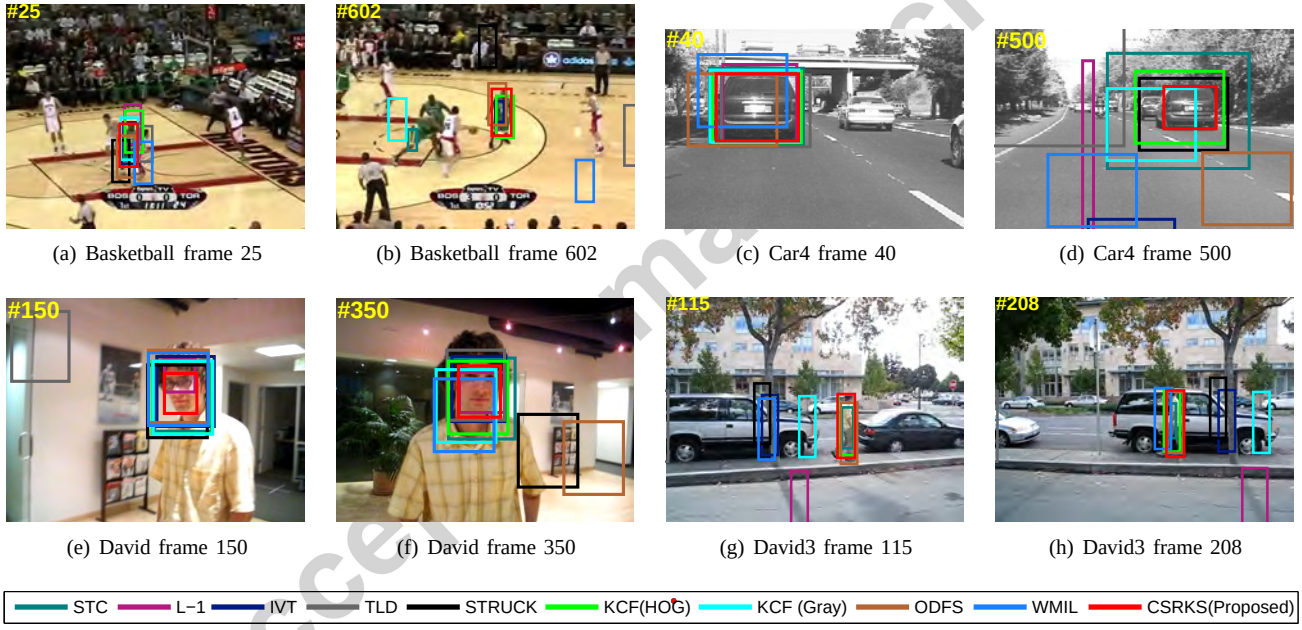
[8] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.



(a) Precision for out of plane rotation

(b) Success for out of plane rotation

Fig. 17. Precision and success plots for out of plane rotation



(a) Basketball frame 25

(b) Basketball frame 602

(c) Car4 frame 40

(d) Car4 frame 500

(e) David frame 150

(f) David frame 350

(g) David3 frame 115

(h) David3 frame 208

Fig. 18. Tracked rectangles in Basketball, Car4, David and David3

[9] C. Zhang, J. C. Platt, and P. A. Viola, "Multiple instance boosting for object detection," in *Advances in neural information processing systems*, 2005, pp. 1417–1424.

[10] O. Maron and T. L. Perez, "A framework for multiple-instance learning," *Advances in neural information processing systems*, pp. 570–576, 1998.

[11] Y. Li, D. M. Tax, R. P. Duin, and M. Loog, "Multiple-instance learning as a classifier combining problem," *Pattern Recognition*, vol. 46, no. 3, pp. 865–874, 2013.

[12] N. C. Oza, "Online ensemble learning," Ph.D. dissertation, Citeseer, 2001.

[13] K. Zhang and H. Song, "Real-time visual tracking via online weighted multiple instance learning," *Pattern Recognition*, vol. 46, no. 1, pp. 397–411, 2013.

[14] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time object tracking via online discriminative feature selection," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4664–4677, 2013.

[15] Y. Xie, W. Zhang, C. Li, S. Lin, Y. Qu, and Y. Zhang, "Discriminative object tracking via sparse representation and online dictionary learning," *IEEE Transactions on Cybernetics*, vol. 44, no. 4, pp. 539–553, 2014.

[16] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A survey of appearance models in visual object tracking," *ACM transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 4, p. 58, 2013.

[17] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

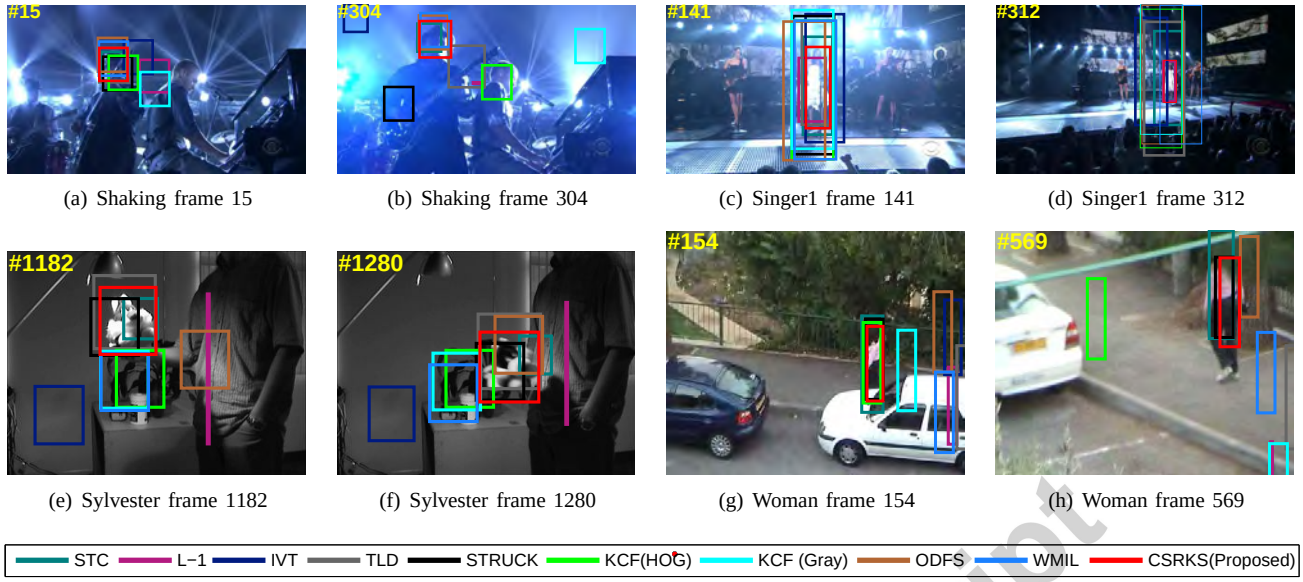
[18] Z. Fan, H. Ji, and Y. Zhang, "Iterative particle filter for visual tracking," *Signal Processing: Image Communication*, vol. 36, pp. 140–153, 2015.

[19] W. Li, P. Wang, and H. Qiao, "Top–down visual attention integrated particle filter for robust object tracking," *Signal Processing: Image Communication*, vol. 43, pp. 28–41, 2016.

[20] Z. H. Khan and I. Y.-H. Gu, "Nonlinear dynamic model for visual object tracking on grassmann manifolds with partial occlusion handling," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 2005–2019, 2013.

[21] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125–141, 2008.

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



(a) Shaking frame 15 (b) Shaking frame 304 (c) Singer1 frame 141 (d) Singer1 frame 312

(e) Sylvester frame 1182 (f) Sylvester frame 1280 (g) Woman frame 154 (h) Woman frame 569

Fig. 19. Tracked rectangles in Shaking, Singer1, Sylvester and Woman

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

[24] T. Bai and Y. F. Li, "Robust visual tracking with structured sparse representation appearance model," *Pattern Recognition*, vol. 45, no. 6, pp. 2390–2404, 2012.

[25] T. Bai, Y.-F. Li, and X. Zhou, "Learning local appearances with sparse representation for robust and fast visual tracking," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 663–675, 2015.

[26] R. T. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1631–1643, 2005.

[27] D. Liang, Q. Huang, W. Gao, and H. Yao, "Online selection of discriminative features using bayes error rate for visual tracking," in *Advances in Multimedia Information Processing-PCM 2006*. Springer, 2006, pp. 547–555.

[28] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[29] S. Avidan, "Ensemble tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 261–271, 2007.

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

[31] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[32] J. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[33] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang, *Fast Visual Tracking via Dense Spatio-temporal Context Learning*. Cham: Springer International Publishing, 2014, pp. 127–141.

[34] F. Yang, H. Lu, and M.-H. Yang, "Robust superpixel tracking," *IEEE Transactions on Image Processing*, vol. 23, no. 4, pp. 1639–1651, 2014.

[35] Y. Yin, D. Xu, X. Wang, and M. Bai, "Online state-based structured SVM combined with incremental PCA for robust visual tracking," *IEEE Transactions on Cybernetics*, vol. 45, no. 9, pp. 1988–2000, 2015.

[36] D. Wang, H. Lu, and C. Bo, "Visual tracking via weighted local cosine similarity," *IEEE Transactions on Cybernetics*, vol. 45, no. 9, pp. 1838–1850, 2015.

[37] X. Zhou, Y. Li, and B. He, "Entropy distribution and coverage rate-based birth intensity estimation in GM-PHD filter for multi-target visual tracking," *Signal Processing*, vol. 94, pp. 650–660, 2014.

[38] X. Zhou, H. Yu, H. Liu, and Y. Li, "Tracking multiple video targets with an improved GM-PHD tracker," *Sensors*, vol. 15, no. 12, pp. 30 240–30 260, 2015.

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

[40] M. Van Breukelen, R. P. Duin, D. M. Tax, and J. Den Hartog, "Handwritten digit recognition by combined classifiers," *Kybernetika*, vol. 34, no. 4, pp. 381–386, 1998.

[41] D. M. Tax, R. P. Duin, and M. Van Breukelen, "Comparison between product and mean classifier combination rules," in *Proc. Workshop on Statistical Pattern Recognition, Prague, Czech*, 1997.

[42] S. Aksoy and R. M. Haralick, "Feature normalization and likelihood-based similarity measures for image retrieval," *Pattern Recognition Letters*, vol. 22, no. 5, pp. 563–582, 2001.

[43] S. Hou and R. C. Qiu, "Kernel feature template matching for spectrum sensing," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 5, pp. 2258–2271, 2014.

[44] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[45] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[46] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Computer vision and pattern recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2411–2418.