Robust Visual Tracking via Multiple Kernel Boosting With Affinity Constraints

Fan Yang, *Student Member, IEEE,* Huchuan Lu, *Senior Member, IEEE,* and Ming-Hsuan Yang, *Senior Member, IEEE*

***Abstract*—We propose a novel algorithm by extending the multiple kernel learning framework with boosting for an optimal combination of features and kernels, thereby facilitating robust visual tracking in complex scenes effectively and efficiently. While spatial information has been taken into account in conventional multiple kernel learning algorithms, we impose novel affinity constraints to exploit the locality of support vectors from a different view. In contrast to existing methods in the literature, the proposed algorithm is formulated in a probabilistic framework that can be computed efficiently. Numerous experiments on challenging data sets with comparisons to state-of-the-art algorithms demonstrate the merits of the proposed algorithm using multiple kernel boosting and affinity constraints.**

***Index Terms*—Affinity constraint, multiple kernel learning, object tracking.**

I. Introduction

VISUAL TRACKING has been one of the fundamental problems in computer vision with numerous applications. Essentially, the task of visual tracking is to determine the object states, such as position, velocity, scale, and other related information, from images. However, the appearance of a target object often changes significantly due to numerous factors (e.g., pose, lighting, and shape deformation) and thus makes visual tracking challenging. Other factors such as occlusion, complex motion, and background clutters further complicate this task. Numerous algorithms have been developed to address these problems via template matching [1]–[7], state estimation [8], [9], as well as foreground and background classification [10]–[17].

Manuscript received December 11, 2012; revised April 9, 2013 and June 2, 2013; accepted June 14, 2013. Date of publication July 31, 2013; date of current version February 4, 2014. The work of F. Yang and H. Lu was supported by the Natural Science Foundation of China under Grants 61071209 and 61272372. The work of M.-H. Yang was supported in part by the NSF CAREER under Grant 1149783 and in part by the NSF IIS under Grant 1152576. This paper was recommended by Associate Editor H. Wang.

F. Yang is with the School of Information and Communication Engineering, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, China, and also with the Department of Computer Science, University of Maryland, College Park, MD 20742 USA (e-mail: fyang@umiacs.umd.edu).

H. Lu is with the School of Information and Communication Engineering, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, China (e-mail: lhchuan@dlut.edu.cn).

M.-H. Yang is with the Department of Electrical Engineering and Computer Science, University of California, Merced, CA 95344 USA (e-mail: mhyang@ucmerced.edu).

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/TCSVT.2013.2276145

For template matching, the reference model can be based on, among others, intensity values, statistical distribution of features, and low-dimensional subspace representations from the image containing the target object. Lucas and Kanade [1] used fixed templates and optical flow techniques to determine the motion of an object. Black and Jepson [2] proposed the eigentracking algorithm using a pretrained view-based eigenbasis representation of the object within the optical flow framework. For an online update of appearance models, Ross *et al.* [4] introduced an incremental subspace learning algorithm for visual tracking. However, this method is still limited in handling occlusion or nonrigid deformation due to the use of a subspace representation where objects are represented with rectangular image patches. Comaniciu *et al.* [3] used a nonparametric distribution to represent objects and estimate of mode shift for tracking. Shen *et al.* [18] extended the work in [3] by introducing kernel SVM into mean shift tracking to incorporate online template update. While existing methods based on template matching have demonstrated success in visual tracking, they are less effective when the object appearance changes significantly.

To better estimate the states without Gaussian distribution assumptions, Isard and Blake [8] introduced particle filters to visual tracking with the condensation algorithm. Furthermore, Pérez *et al.* [9] proposed an adaptive algorithm that integrates color distributions with particle filters and adapts to the object appearance change. Nevertheless, this tracker accumulates drift error during update and often fails in cluttered scenes.

For classification-based tracking algorithms, the task is to discriminate the target object from the background in successive frames. Avidan extends the optical flow framework with support vector machine [10] to classify target objects. However, it requires a large collection of samples for off-line training. In addition, the learned classifier may not be effective to account for all possible changes of object appearance. The ensemble tracking approach [11] combines a number of weak classifiers via boosting to learn a decision boundary for separating target objects from the background, and then iteratively train new weak classifiers for update. Collins *et al.* [12] proposed a method to adaptively select color features that best separate the object from the background. However, the pixel-based representations used in the above-mentioned methods are not effective in handling heavy occlusions and clutters. Grabner *et al.* [13] applied an online Adaboost algorithm to object tracking in which newly

observed samples are used to train new classifiers. Babenko *et al.* [15] proposed a tracking algorithm with an online multiple instance learning (MIL) boosting approach to learn an object detector. More recently, Kwon and Lee [19] developed a visual tracking decomposition (VTD) system that consists of a fixed number of basic trackers emphasizing different features to deal with various scenarios. Li *et al.* [20] constructed an appearance model using the 3-D discrete cosine transform, which generates a compact energy spectrum for object representation, in visual tracking. Although these algorithms are able to deal with certain shape deformation or drifts, they are not effective in tracking objects with large appearance change caused by nonrigid motion or large pose variation.

Recently, an increasing number of studies applied sparse coding to visual tracking and generate state-of-the-art results. [21] simply uses holistic object samples as templates for the dictionary and computes sparse codes by solving ℓ_1 minimization. To improve both effectiveness and efficiency of the tracker proposed in [21] and [22] adopts PCA basis vectors as object templates and presents a fast iterative solution. Liu *et al.* [23] constructed a dictionary using a K-selection approach. Zhong *et al.* [24] combined a sparsity-based discriminative classifier with a generative model based on both holistic and local representations, where spatial information is also encoded. Jia *et al.* [25] proposed an alignment pooling approach to obtain global sparse representations from local patches. The templates are also updated to capture object changes. Zhang *et al.* [26] applied the multitask learning framework using the group sparsity constraints among candidates, where each candidate can be considered one task.

In this paper, we pose visual tracking as a binary classification problem and propose an algorithm to exploit multiple classifiers and features within the multiple kernel learning (MKL) framework. The proposed method aims at finding an optimal combination of hyperplane regarding specific features that can best classify two classes from a large pool of classifiers and image attributes. Rather than combining all features in the same kernel space, we distribute all features to a group of different kernels. The classification performance of features on specific kernel-based classifiers can thus be thoroughly evaluated, which leads to a more discriminative ability to the final decision function. Moreover, we utilize the boosting technique for efficiently selecting good hyperplanes from support vector machines (SVM) to reduce the computation load that is different from related methods based on global optimization. It is worth noticing that our method differs from other boosted MKL algorithms in several aspects. The formulation in this paper does not require solving an optimization problem as the LPBoost algorithm that entails joint optimization of parameters (LP-β) or large training data [27]. On the other hand, the boosting approach with MKL in [28] is designed to choose good features in which kernels are treated as elements in a feature vector and only one SVM classifier is used. Thus, it can be considered dimensionality reduction rather than combination of classifiers and features as formulated in this paper. In fact, reference [28] treats all kernel functions as a single feature and only uses one SVM classifier, whereas the proposed multiple kernel boosting (MKB) algorithm treats kernel functions separately and uses multiple SVM classifiers. The proposed multiple kernel boosting algorithm facilities efficient update of classifiers to account for appearance change in object tracking. Finally, we exploit affinity constraints to exploit the locality of data for learning robust classifiers. Experimental results demonstrate that our algorithm is robust in complex scenes with favorable performance over existing methods.

Preliminary results of this paper were presented in [29] and further improved in this paper. First, we compare the proposed work with more related algorithms. In particular, we explicitly differentiate our MKB tracker from multiview learning and other feature integration approaches applied to visual tracking, thereby providing more justifications of this paper. Second, we clarify the difference between the proposed locality affinity constraints and the standard localized multiple kernel learning algorithm that also take spatial information into account. Third, we increase the number of combinations of kernels and features by adjusting the parameters of kernels to a wider range. Therefore, we increase the variety of the combinations and strengthen the capability of the tracker in capturing more various changes of the objects. Finally, we enrich the experimental validation. In [29], only seven sequences are tested, among which four are from the same dataset that only contains moving persons. To better convince the reader that our MKB tracker is able to handle various situations, we conduct experiments on sequences containing more various and complex scenarios. To demonstrate that the proposed MKB tracker is not a direct application of the canonical MKL algorithms to visual tracking domain, we conduct additional experiments by directly incorporating a standard MKL algorithm and a localized MKL algorithm into the tracking framework without any other changes. Experimental results clearly show that our MKB tracker performs much better than the simple MKL-based trackers.

II. Related Work and Context

There is a rich amount of literature on the application of effective classifiers such as SVM for object tracking [10]. However, these methods typically do not involve kernel design even though numerous studies have shown that it is of prime importance for improving accuracy of SVM classifiers. In this paper, we explore kernel design within the multiple kernel learning framework [30], [31], which has shown great promise in recent object classification tasks, for visual tracking. The MKL algorithms aim to compute an optimal combination of weighted kernels in supervised learning paradigm. Rather than using one single kernel, the MKL algorithms fuse different features and kernels in an optimal setting. Recently, numerous methods have been proposed within this learning framework. Rakotomamonjy *et al.* [31] proposed the SimpleMKL method for simplifying the optimization process based on mixed-norm regularization. The localized MKL [32] and Bayesian localized MKL [33] methods exploit the distribution of training data in each kernel space and assign higher weights to appropriate kernel functions. On the other hand, Cao *et al.* [34] proposed the heterogeneous feature machines

to learn a nonlinear combination of multiple kernels by utilizing group LASSO constrains. Yang *et al.* [35] proposed group-sensitive multiple kernel learning to accommodate intraclass diversity and interclass correlation for object categorization. In addition, the boosting method has also been incorporated into the MKL framework for feature combination [27] and selection [28].

Another related line of research is the multiview learning that aims to exploit multiple representations from different independent sets of features. Blum and Mitchell [36] proposed the cotraining framework to train two learners with different representations of the labeled samples to iteratively help each other to label the unlabeled samples. Abney [37] presented a theoretical analysis of cotraining with an upper bound of the error rate based on certain independence assumptions. In addition, other methods and applications have been proposed, including the coregularization [38], active learning [39], clustering [40], and object tracking [41]. The difference between the MKL algorithm and the multiview learning lies mainly in two aspects. First, multiview learning algorithms require sufficient and conditionally independent feature sets. But, there is no such assumption for the MKL algorithms. Second, multiview learning methods need to deal with view disagreement [42]. However, the MKL algorithm learns a single decision function from the combination of multiple kernel functions instead of learning separate decision functions as in the multiview learning.

Furthermore, numerous feature integration algorithms for visual tracking have been proposed. Yin *et al.* [43] proposed a generic likelihood map fusion framework to combine different features to generate a set of likelihood maps where objects are segmented based on their confidence scores. In [44], multiple cues are combined to enhance the discriminative capability of the joint observation model in its neighborhood, which is achieved by solving a regression problem. Lu *et al.* [45] combined color and texture features to build a pixel-wise spatial pyramid for robust tracking. Similarly with [43], Wang *et al.* [46] developed a set of probability maps based on different cues and combined them by a weighted linear function where the weights of probability maps are updated dynamically. In [47], multiple trackers with different features are utilized to cope with different challenges. To fuse independent trackers, two kinds of configurations are proposed for tracker selection and tracker interaction in a Bayesian framework. However, this paper focuses more on the tracker integration than feature combination.

In this paper, we propose an MKB algorithm with affinity constraints for robust visual tracking. To describe an object, we use three types of feature descriptors, including RGB histogram, histogram of oriented gradients (HoG) [48], and SIFT descriptors [49]. Each sample is mapped to four kernel spaces with linear, polynomial, RBF, and sigmoid mapping functions. For each SVM classifier, we use one kernel function to map one type of feature. The parameters of these kernels are varied within a range, thereby resulting in a pool of SVM classifiers representing multiple combinations of kernels and features. To find an optimal collection of combinations under a specific scenarios, we formulate this task with boosting algorithms rather than a time-consuming optimization on the whole training set as used in existing MKL algorithms [30]. That is, we consider each single kernel SVM classifier as a weak classifier and determine an optimal combination via boosting. We note that the proposed MKB algorithm significantly outperforms standard MKL methods [30] in terms of computational complexity. Moreover, different from the ensemble learning utilizing homogeneous weak classifiers, the MKB method uses heterogeneous weak classifiers as each one deals with one type of feature. In other words, the ensemble learning method aims at constructing a strong classifier from only one view, while the MKB method achieves this goal from multiple views as we exploit the data from multiple kernel mapping spaces. In addition, we also exploit locality information of input data via a probabilistic model in the classifier design. The locality information imposed on the decision function measures the similarity or affinity between training samples and test data. For online update, we retrain the set of single kernel SVM classifies, select some discriminative ones by the MKB algorithm, and compute the distribution of support vectors to obtain new locality affinity constraints.

III. Multiple Kernel Boosting

A. Multiple Kernel Learning

Support vector machines have been successfully applied to numerous classification and regression problems. One important issue in such tasks is to select an appropriate data representation. In SVM-based methods, the data representation is implicitly chosen by the kernel function $K(x, x_i)$, where $K(\cdot, \cdot)$ is a given positive definite function associated with a reproducing kernel Hilbert space. However, it is difficult for a single SVM classifier to choose a good kernel for the given training dataset in some cases. To address this problem, the MKL framework [30] is proposed which has been shown to enhance the interpretability of the decision function and improve classification performance. Specifically, the MKL algorithm aims to find an optimal convex combination of multiple basis kernels and the associated classifier simultaneously. Here, we use a binary classification to explain the principle of the MKL algorithm. Given a set training samples $\{x_i, y_i\}_{i=1}^{D}$, where x_i is the ith sample and $y_i = \{\pm 1\}$ indicates the corresponding label, the task is to train a multikernel based classifier $F(x)$ to classify unlabeled samples. Let $\{K_m\}_{m=1}^{M}$ be the kernel set. The combination of multiple kernels is defined as

$$K(x, x_i) = \sum_{m=1}^{M} \beta_m K_m(x, x_i) \tag{1}$$

where kernel weights $\beta_m \geq 0$ and $\sum_{m=1}^{M} \beta_m = 1$. In this formulation, K_m can be the same classical kernels with different hyperparameters or different kernels. In addition, they can be applied to different feature sets. The main task of the MKL algorithm is to compute the weights $\{\beta_m\}_{m=1}^{M}$ of these kernels. Then, the decision function is defined as

$$F(x) = \sum_{i=1}^{D} \alpha_i y_i \sum_{m=1}^{M} \beta_m K_m(x, x_i) + b \tag{2}$$

where $\{\alpha_i\}_{i=1}^{D}$ and b are the Lagrange multipliers and the bias in the standard SVM algorithm. The parameters $\{\alpha_i\}_{i=1}^{D}$, $\{\beta_m\}_{m=1}^{M}$ and b can be optimized jointly [30].

B. Multiple Kernel Boosting

Despite its success in applications such as category-level object recognition, the MKL methods cannot be directly applied to visual tracking due to several reasons. First, the optimization process requires a large amount of training samples for convergence, which is not the case for efficient object tracking. Second, kernel weights are fixed after training for applications such as category-level object recognition. However, object tracking entails adaptive weights for classifiers to effectively differentiate foreground targets from background clutters. Although the SimpleMKL [31] model modifies the optimization process of classical MKL methods to improve efficiency, the computational complexity is still considerable. Recently, Gehler and Nowozin [27] presented a boosting algorithm based on MKL for feature combination and demonstrated that the proposed LPBoost method performs well in terms of speed with few training samples. Siddiquie *et al.* [28] applied the GentleBoost algorithm to MKL for combining multiple features. Their method learns a mixture of kernels by greedily selecting exemplars corresponding to each kernel. However, existing boosting extensions of MKL are not effective for object tracking as they are developed to enhance accuracy of category-level object recognition for a large set of instances (i.e., generalization) rather than localization of a specific object (i.e., specificity), as borne out by numerous experiments with different image sequences in Section VI. We propose a multiple kernel boosting algorithm for feature selection for object tracking to reduce computational load and increase empirical accuracy.

For a sample x, we construct a vector by concatenating its kernel values to all the other training samples $\{x_i, y_i\}_{i=1}^{D}$ to indicate the mth kernel response

$$K_m(x) = [K_m(x, x_1), K_m(x, x_2), ..., K_m(x, x_D)]^\top \tag{3}$$

and rewrite (2) as

$$\begin{aligned} F(x) &= \sum_{m=1}^{M} \beta_m \sum_{i=1}^{D} \alpha_i y_i K_m(x, x_i) + b \\ &= \sum_{m=1}^{M} \beta_m (K_m(x)^\top \alpha + b_m) \end{aligned} \tag{4}$$

where $\alpha = \left[\alpha_1 y_1, \alpha_2 y_2, ..., \alpha_D y_D\right]^\top$ and $b = \sum_{m=1}^{M} b_m$. The standard MKL formulation is converted into a linear combination of the real outputs of M separate SVM classifiers $K_m(x)^\top \alpha + b_m$. We train each of the M SVM classifiers with different parameters $\{\alpha_m, b_m\}$ first, and then optimize the weights $\{\beta_m\}_{m=1}^{M}$. Each individual SVM classifier is not restricted to share the same parameter space. Let h_m denote the response of each SVM classifier, $h_m(x) = K_m(x)^\top \alpha_m + b_m$. Note that each kernel function has an individual α coefficient in our boosting-based MKL algorithm, which is different from the standard MKL algorithms where all kernel functions share a single α. Instead of computing $\{\beta_m\}_{m=1}^{M}$ with optimization techniques, we formulate this problem with the proposed multiple kernel boosting algorithm. The decision function of this boosting algorithm is

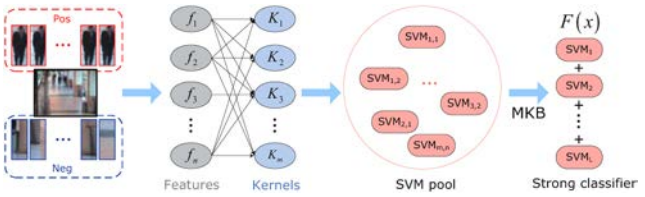


Fig. 1. Multiple kernel boosting (MKB) process. For each training image, multiple features from positive samples and negative samples are extracted and put into multiple kernel functions to form a pool of SVM classifiers. Each SVM classifier focuses on a specific combination of feature and kernel. Performing boosting on the SVM classifier pool, we obtain a strong classifier consisting of multiple weak SVM classifiers.

$$F(x) = \sum_{l=1}^{L} \beta_l h_l(x) \tag{5}$$

where L indicates the total number of iterations. The notation m is replaced by l to indicate the index of iteration rather than the kernel function. From this perspective, we regard the MKL method as choosing multiple weak single kernel SVM classifiers to form a strong one. Our MKB method does not entail solving a complex optimization program, thereby facilitating an efficient and effective algorithm for tracking. Furthermore, this formulation easily accommodates an online update module (Section V) for robust tracking.

As Fig. 1 shows, we extract a set of features $\{f_1, f_2, ..., f_N\}$ from the positive and negative examples of the training set for selecting a set of $\{K_1, K_2, ..., K_M\}$ kernels, thereby obtaining a pool of $M \times N$ combinations of kernel machines. For each combination, we train a single kernel SVM classifier in the respective input feature space. The weighted classification error of a single SVM classifier is defined as

$$\varepsilon = \frac{\sum_{i=1}^{D} w(i) \cdot |h(x_i)| \cdot U(-y_i h(x_i))}{\sum_{i=1}^{D} w(i) \cdot |h(x_i)|}. \tag{6}$$

Here, $U(x)$ is a function that equals 1 when $x > 0$ and 0 otherwise, $w(i)$ is the weight of the training samples, and $h(x_i)$ is the real-valued classification output of the SVM classifier on the input x_i. From the $M \times N$ weak classifiers, we adaptively select multiple features and kernels to form an optimal strong classifier.

The main steps of the boosting process are as summarized in Algorithm 1. Given the training sets, we first train a single kernel SVM classifier for each combination of features and kernel functions, thereby constructing a pool of candidate single kernel SVM classifiers. First, we initialize equal weights for training samples. During each iteration l, we select an SVM classifier that gives the smallest weighted classification error on the entire dataset, and then compute the weight for this selected SVM classifier. If the weight is smaller than 0, the procedure terminates since even the best SVM classifier performs worse than guessing; otherwise, the selected classifier

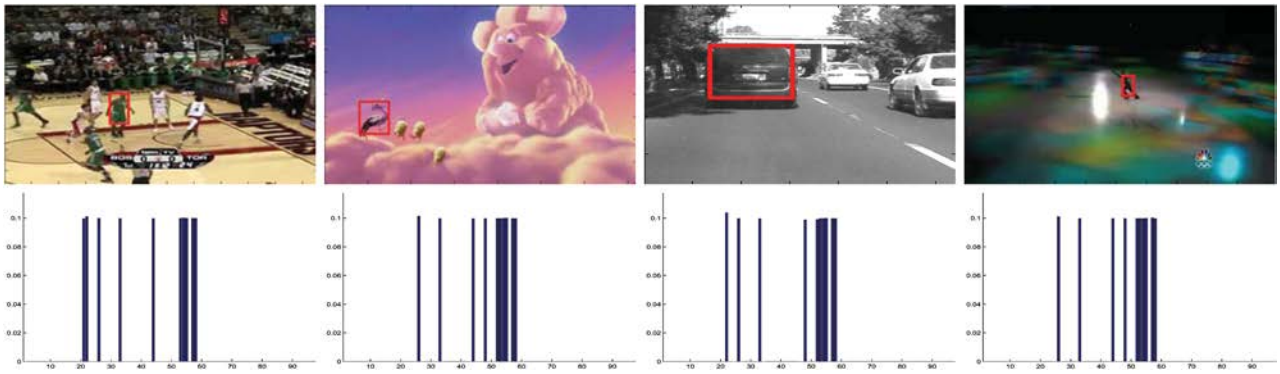


Fig. 2. Weights of selected SVM classifiers after the MKB process on four representative sequences. First row: the first frames with initial bounding boxes enclosing the target to be tracked. Second row: the weight distribution of the selected SVM classifiers for each sequence. The x-axis indicates index of weak SVM classifiers (96 of them are used in our implementation), and the y-axis denotes the normalized weight. Note that the weights are different for different scenarios although there are some classifiers with strong discriminative strength often selected by the MKB algorithm.

Algorithm 1 Multiple Kernel Boosting (MKB)

Input: Training sets $\{x_i, y_i\}_{i=1}^{D}$, feature functions $\{f_n\}_{n=1}^{N}$, kernel functions $\{K_m\}_{m=1}^{M}$, the decision function $F(x) = 0$

1: For each $n \in N$ and $m \in M$, train a single kernel SVM $h_{m,n}(x)$ on feature f_n and kernel K_m on the entire training set $\{x_i, y_i\}_{i=1}^{D}$ to form a pool of candidate single kernel SVMs, denoted as h

2: Initialize sample weights $w_1(i) = 1/D$

3: For $l = 1$ to L do

1) For each $h_{m,n}(x)$, compute weighted classification error $\varepsilon_{m,n}$ using (6)

2) Select $h_l(x) = \arg\min_{h_{m,n} \in \mathrm{h}} \varepsilon_{m,n}$

3) Compute weight $\beta_l = \frac{1}{2} \log \frac{1-\varepsilon_l}{\varepsilon_l}$ for $h_l(x)$

4) If $\beta_l < 0$, break; otherwise add $h_l(x)$ to the decision function $F(x) \leftarrow F(x) + \beta_l h_l(x)$

5) $w_{l+1}(i) = \frac{w_l(i)}{Z_l} e^{-\beta_l y_i h_l(x_i)}$ where Z_l is a normalization factor and $h_l(x_i)$ is the classification output (1 or -1) of classifier h_l on sample x_i

4: End

Output: MKB classifier $F(x) = \sum_{l=1}^{L} \beta_l h_l(x)$

is added to the decision function. Then, we update the weights for the training samples where samples incorrectly classified are assigned larger weights in the next iteration. Finally, the algorithm outputs a strong classifier consisting of a number of single kernel SVM classifiers.

We note that the proposed MKB algorithm is not a variant of the previous ensemble tracking algorithm [11] where all weak classifiers are homogeneous in the same feature space. Although increasing the number of classifiers is likely to improve the classification performance, essentially the data is represented within one specific feature space and thus the discriminability of the final ensemble classifier is limited. In contrast, the MKB method trains a collection of classifiers constructed with different features. By using the kernel techniques, these classifiers operate in multiple feature spaces, and thus likely provide complementary representations of the same data. Even when not all kernels perform well in a specific input data space, the MKB method still works well as long as at least one kernel has sufficiently discriminative ability. The MKB method does not introduce new classifiers or discard old ones, but selects the most effective kernel machines in an active set. As tracking proceeds, some kernel machines may be deactivated or reactivated to deal with specific scenarios. In addition, the number of selected SVM classifiers may vary after each update because the boosting process may choose more classifiers when the object becomes difficult to track, whereas both the VTD and ensemble tracking methods use a fixed number of basic trackers that are less adaptive although they also train a series of classifiers with different features. With the boosting process, an effective machine can be assembled from multiple kernels efficiently to account for different tracking scenarios.

Fig. 2 illustrates the weights of selected SVM classifiers after the MKB procedure.

We use three features (64-dimensional RGB histograms, 128-dimensional HoG, and 128-dimensional SIFT descriptors), and four kernels (linear, polynomial, RBF, and sigmoid functions) with varying parameters, resulting in 96 SVM classifiers.

The results show that different combinations of kernels and features are selected with different weights for effective visual tracking in different scenes.

IV. Affinity Constraints

We further improve the MKB algorithm by exploiting the local distribution of training data. Similar concepts have been exploited in [32] where a localized MKL algorithm is proposed by adding a gating function to determine the kernel weight according to the input sample. The core idea is that the correlation of input data and a specific kernel function affects the weight of the kernel. Therefore, the weight varies based on the correlation of input sample with the training data. Likewise, Yang *et al.* [35] extended the MKL method by exploiting groups of similar data with group-sensitive kernel weights. In this paper, we propose to incorporate the distribution of

training data into $F(x)$ to enhance the robustness of the MKB algorithm. We rewrite (2) as

$$F(x) = \sum_{i=1}^{D} \alpha_i y_i \sum_{m=1}^{M} \beta_m(x) K_m(x, x_i) + b \tag{7}$$

where $\beta_m(x)$ is a gating function of input x rather than a constant β_m in the standard MKL method. An example of softmax $\beta_m(x)$ in [32] and [35] is defined as

$$\beta_m(x) = \frac{\exp(K_m(v_m, x) + v_{m0})}{\sum_{k=1}^{M} \exp(K_k(v_k, x) + v_{k0})} \tag{8}$$

where v_m and v_{m0} are the parameters of the gating function and the normalized exponential formation guarantees the nonnegativity of $\beta_m(x)$ which can be learned iteratively [32]. Recently, the sigmoid and Gaussian gating models are also introduced in [50] with similar optimization approach but with adaptive step size during iteration to achieve better convergence. Since multiple features of different dimensions are used, it is not feasible to adopt (8) as the sizes of the input samples are different. Moreover, the optimization process is rather time-consuming [35], [50], and the optimization results are sensitive to initial values. In this paper, we present a simple yet effective method to exploit the underlying distribution of training data for visual tracking. We note the numerator of (8), $\exp(K_m(v_m, x) + v_{m0})$, can be considered the exponential form of the classification score from an SVM classifier with kernel K_m on the input sample x. It reflects the confidence of the mth kernel of classifying the sample x as positive sample, and the affinity of x to the positive samples with respect to the kernel K_m. That is, (8) can then be explained as the importance of kernel K_m among all kernels to input x. Therefore, we can exploit this affinity information and ensure efficiency at the same time.

It is clear that individual SVM classifiers trained in the MKB algorithm have recorded the distribution of training data on respective features and kernels. Since support vectors of each SVM classifier determine the classification margin and capture the essential information of training data, we utilize them for exploiting the locality of data. We factor the weight β_l of (5) into two terms, β_l^* and $A_l(x)$, and have

$$F(x) = \sum_{l=1}^{L} \beta_l^* A_l(x) h_l(x) \tag{9}$$

where β_l^* is the same as β_l of (5) that is computed by the MKB algorithm. $A_l(x)$ is a function of input x that indicates the similarity of x with the trained SVM classifier and is referred to as affinity constraint in our algorithm. If an input sample has high affinity with the distribution of support vectors in a specific SVM classifier, the importance of the corresponding SVM classifier is high and assigned with larger weight. It has been shown that the distributions of support vectors can be approximated by Gaussian distribution and thereby solved effectively and efficiently [51].

We model the affinity constraint of each sample based on its corresponding probabilistic distribution

$$A_l(x) = 1 - \exp(-|\sigma_l(x)|) \tag{10}$$

with log odds ratio $\sigma_l(x) = \log\left[\frac{p_l(y=1|x)}{p_l(y=-1|x)}\right]$ and we model the distribution of support vectors with Gaussian distributions $p_l(x|y=1) \sim N(\mu_1, \sigma_1)$ and $p_l(x|y=-1) \sim N(\mu_{-1}, \sigma_{-1})$. For each trained SVM classifier $h_l(x)$, we compute the mean μ_l^+ and μ_l^- of positive and negative support vectors respectively. Assuming uniform prior for both classes, the posterior x belonging to each class is

$$\begin{aligned} p_l(y=1|x) &= \exp(-\left|x - \mu_l^+\right|) \\ p_l(y=-1|x) &= \exp(-\left|x - \mu_l^-\right|). \end{aligned} \tag{11}$$

As the value of the affinity is between 0 and 1, it can be seen as the probability of the sample x belonging to the support vectors. If x is similar to the training data on a specific combination of feature and kernel, the importance of the corresponding SVM classifier is high, and vice versa. Therefore, we formulate the distribution of training samples and impose this affinity constraint on testing samples to improve the discriminative ability of the decision function. It is noteworthy that in our method, we do not recourse to optimization (as done in [32]) in the training procedure. Instead, we only need means of positive and negative support vectors, μ_l^+ and μ_l^-, after we have trained SVM classifiers. This approach also makes update of the affinity constraint simple and quick by computing only μ_l^+ and μ_l^-. Meanwhile, the computational complexity of our constraints is much lower than that of methods using gating function and fits well for tracking tasks.

V. Tracking via MKB and Affinity Constraints

In this section, we introduce an effective visual tracking algorithm based on the proposed multiple kernel boosting with affinity constraints. The object motion is modeled by similarity transform with four parameters in this paper although the affine model can also be adopted. In the first frame, the initial state of the target object is manually initialized with $x^1 = (c_x^1, c_y^1, s^1, \theta^1)$ that represents the corresponding position, size, and rotation angle (the superscript denotes the current frame number). To increase the number of training samples, we crop out a set of images $X^+ = \{x_i | 0 \le l(x_i) - l(x^1) < r_\alpha\}_{i=1}^{D^+}$ to collect positive samples where r_α is a small constant and $l(x)$ indicates the center of x. Similarly, we collect a set of negative samples $X^- = \{x_i | r_\beta \le l(x_i) - l(x^1) < r_\gamma\}_{i=1}^{D^-}$ where r_β is larger than r_α and allows less than one quarter region overlap between positive and negative samples. For concreteness, we use visual tracking as one application of the proposed algorithm and thus use the RGB, HoG and SIFT features to represent objects using the MKB algorithm with affinity constraints.

Given all the image observations up to frame t, $z^{1:t-1} = \{z^1, z^2, ..., z^{t-1}\}$, the state x is predicted by Bayesian filtering

$$p(x^t|z^{1:t}) = \frac{p(z^t|x^t)p(x^t|z^{1:t-1})}{p(z^t|z^{1:t-1})} \tag{12}$$

where $p(x^t|z^{1:t-1}) = \int p(x^t|x^{t-1})p(x^{t-1}|z^{1:t-1})dx^{t-1}$ and $p(z^t|x^t)$ is the observation likelihood. The posterior probability $p(x^t|z^{1:t})$ is approximated by D particles $\{x_i^t\}_{i=1}^D$ with importance weight w_i^t, which are drawn from a reference distribution

Algorithm 2 MKB Tracking with Affinity Constraints

Input: Training sets $\{x_i, y_i\}_{i=1}^{D}$, feature sets $\{f_n\}_{n=1}^{N}$, kernel functions $\{K_m\}_{m=1}^{M}$, and the decision function $F(x) = 0$
Output: Tracking results in each frame $\{x^1, x^2, ..., x^t\}$
Initialization I_t **(**$t = 1$**)**
1: Given the initial state $x^1 = (c_x^1, c_y^1, s^1, \theta^1)$, extract D^+ positive samples and D^- negative samples
2: Extract features $\{f_n(x^1)\}_{n=1}^{N}$ from x^1 and train individual single kernel SVM classifiers $h_{m,n}(x)$
3: Compute affinity function $A_{m,n}(x)$ according to the distribution of support vectors for each trained SVM classifier $h_{m,n}(x)$
4. Apply **Algorithm 1** to construct a strong classifier $F(x)$
For each new frame I_t **(**$t > 1$**)**
1: Sample D particles $\{x_i^t\}_{i=1}^{D}$ around the tracked object x^{t-1} according to distribution $p(x^t|x^{t-1})$. The weight of each particles $\{w_i^t = 1\}_{i=1}^{D}$
2: Use $F(x)$ to obtain classification results of $\{x_i^t\}_{i=1}^{D}$, then $\{w_i^t = \exp(F(x_i^t))/Z^t\}_{i=1}^{D}$, where Z^t is a normalized value
3. Determine the tracking result by $x^t = \sum_{i=1}^{D} w_i^t x_i^t$
4: Treat x^t as positive sample and collect negative samples around x^t, push them into the sample queue Q
5: If the sample queue is full
1) Select SVM classifier $h_{m,n}(x)$ from weak SVM classifier pool h, extract feature $S_Q = f_n(x), x \in Q$. Update $\mu_{m,n}^+$ and $\mu_{m,n}^-$ of support vectors to obtain new $A_{m,n}(x)$
2) Perform **Algorithm 1** again to reselect appropriate $h_l(x)$ to construct a new $F(x) = \sum_{l=1}^{L} \beta_l^* A_l(x) h_l(x)$
3) Empty the sample queue Q
6: Output x^t and proceed to the next frame

$q(x^t|x^{1:t-1}, z^{1:t})$. The reference distribution is approximated by $p(x^t|x^{t-1})$, then the weights can be computed by $w_i^t = w_i^{t-1} p(z^t|x_i^t)$. The state transition, $p(x^t|x^{t-1})$, is modeled by a random walk with a diagonal Gaussian distribution. At frame I_t, we have D candidates with different samples around previous state x^{t-1} from frame I_{t-1}. We use the MKB classification as our likelihood model, $p\left(z^t|x_i^t\right) = e^{F(x_i^t)}$ and determine the particle weights. The weights, $\{w_i^t\}_{i=1}^{D}$, are normalized, and the object state is computed by the weighted sum of particles $x^t = \sum_{i=1}^{D} w_i^t x_i^t$.

To account for appearance change, we incorporate an update scheme in our tracking algorithm. In each frame, we consider the tracked object at the current state x^t as the positive sample, and extract four negative samples from four directions (up, down, left, right) without overlapping regions. The samples from the four directions consist mostly of the background and are thus discriminative enough for distinguishing the foreground object from the background. We accumulate these samples for a few frames (e.g., five or ten frames in this paper) in a queue Q and have a sufficient number of new samples for retraining individual SVM classifiers. Consequently, we obtain updates of $A_l(x)$ and $F(x)$ to account for the most recent appearance change of target objects. The main steps of our tracking algorithm via MKB and affinity constraints are summarized in Algorithm 2.

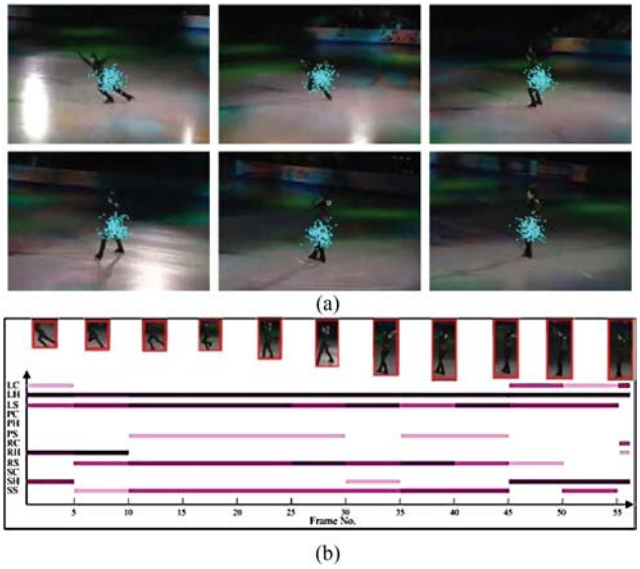


Fig. 3. Particle sampling and classifier update. (a) Colored dots indicate the centers of the candidates proposed by the particle sampling. (b) Twelve SVM classifiers are used and top five with largest weights at each frame are shown. Darker bars indicate higher weights. The x-axis represents the index of frame and the y-axis represents the 12 SVM classifiers. Examples of tracking results are also shown. See the text for details.

Fig. 3 illustrates the particle sampling and the update process. The colored dots in the figure indicate the drawn particles, which correspond to the centers of likely observations in the current frame. Some representative tracking results and update process are also presented where the x-axis denotes frame index as well as the y-axis denotes individual SVM classifiers. The first letters L, P, R, and S are shorthands for linear, polynomial, RBF, and sigmoid kernels. Similarly, the second letters C, H, and S are shorthands for color, HoG, and SIFT feature descriptors. For clarity, we only present five SVM classifiers with the largest weights as indicated by the shades of the bars (darker bars represent heavier weights). The tracking algorithm is updated every five frames and the chart shows that different SVM classifiers (constructed based on different combinations of kernels and features) and their weights are selected to account for appearance change.

VI. Experiments

A. Experimental Settings

The proposed tracking algorithm is implemented in MATLAB and has been made available to the public.[1] We use three features (RGB histogram, HoG, and SIFT descriptors) and four types of kernels (linear, polynomial, RBF kernel, and sigmoid functions) to represent objects. Specifically, we use five different degrees and five different offsets for the polynomial kernels. Similarly, five offsets are used in the sigmoid kernels. For the linear and RBF kernels, the parameters are

[1]Available at http://ice.dlut.edu.cn/lu/publications.html, http://www.umiacs.umd.edu/ fyang, http://faculty.ucmerced.edu/mhyang/pubs.html.

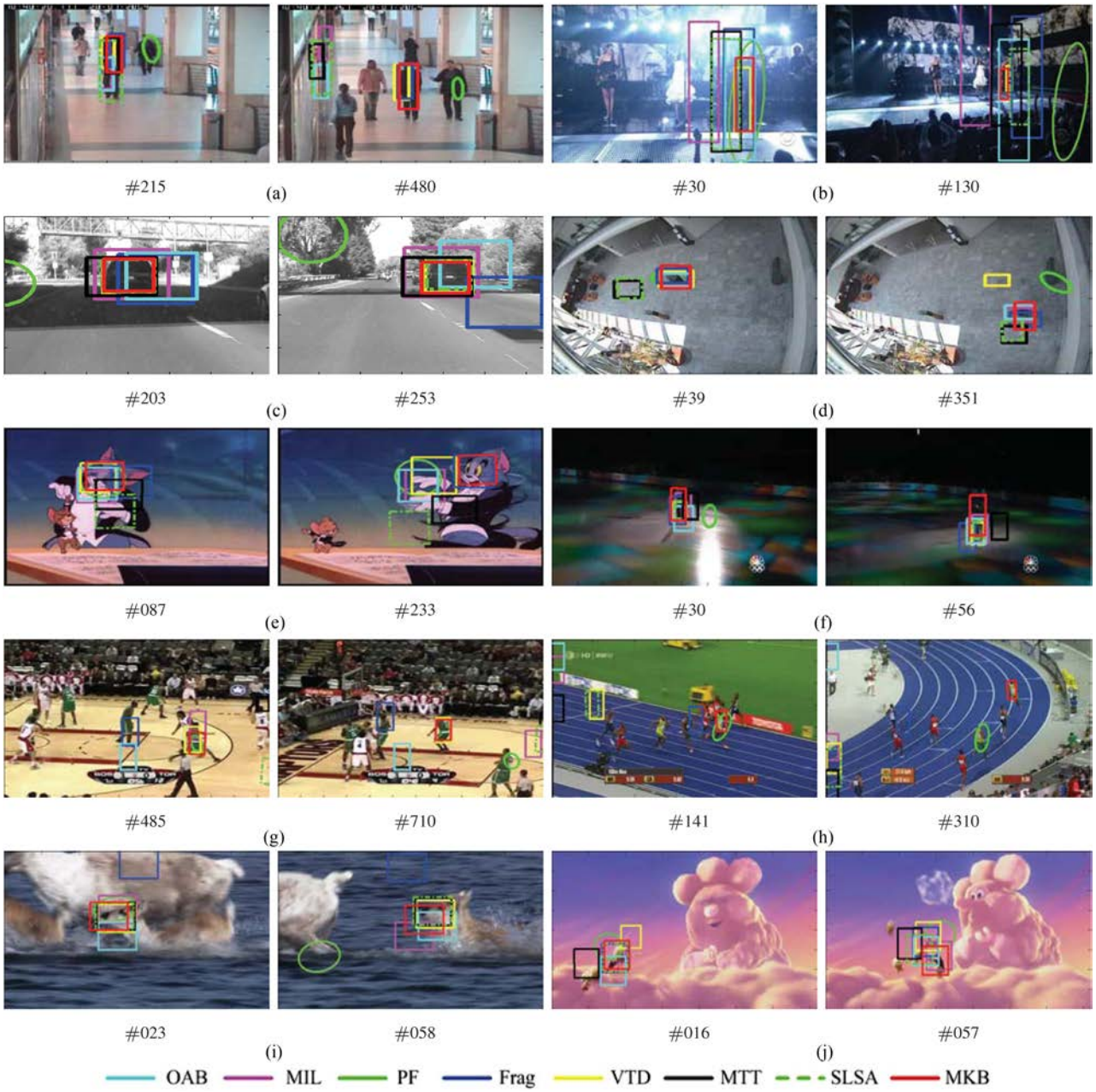


Fig. 4. Qualitative evaluation of different tracking algorithms. (a) *Caviar 1*. (b) *Singer*. (c) *Car*. (d) *Caviar 2*. (e) *tom1*. (f) *Skate*. (g) *Basketball*. (h) *Bolt*. (i) *Animal*. (j) *bird2*.

determined based on the work of Keerthi and Lin [52]. With three different features, we obtain 96 combinations of kernels and features. The iteration number of the MKB algorithm is set to 10, while the number of selected SVM classifiers varies according to different scenarios. Only 200 particles are drawn in each frame and the MKB function $F(x)$ is updated every five or ten frames according to different scenarios.[2]

We evaluate our algorithm with five state-of-the-art trackers, including the online Adaboost tracking method (OAB) [13], the color-based particle filter tracking algorithm (PF) [9], the FragTrack algorithm [53], the MIL method [15], the VTD system [19], the tracker by structured local sparse appearance model (SLSA) [25], and the multitask sparse learning tracking (MTT) [26]. Similar to our MKB tracking method, both the OAB and MIL tracking methods rely on boosting techniques and use new samples to update weak classifiers and corresponding weights. In our experiments, the number of selectors in the OAB method is empirically set to 100. The PF tracking method uses the RGB color space and its sampling parameters are fixed for all sequences. The search region of the FragTrack method is set to 10×10 pixels, and the number of bins of gray-scale intensity histograms is 16. For the VTD method, the number of candidates in each iteration is 100 for

[2]Note that all other parameters are fixed in all the experiments except the parameters for particle filters.

all sequences. For the MTT and SLSA methods, we use the default parameters used in the papers.

We use ten sequences (eight are publicly available and two are from our own dataset) to evaluate the proposed MKB tracking algorithms with other methods. These sequences include challenging factors for object tracking such as occlusion, fast motion, large change of pose and scale, illumination change, as well as complex background.

The computational complexity of the proposed tracking framework depends on the number of sampled candidates and the selected weak SVM classifiers. Suppose the MKB algorithm selects m SVM classifiers after training and randomly samples n candidates for each frame during tracking, the computational complexity for each frame is $O(mn)$. Although there are nearly 100 SVM classifiers, m is usually small so our algorithm is still efficient. During update, we only have 25 samples for RGB feature and HoG feature and no more than 100 samples for SIFT feature to retrain the SVM classifiers. Due to the low dimensionality of features and small number of new samples, the training is very efficient, which generally takes less than 1 s. The HoG and SIFT feature extraction is done by the VLFeat library [54], which is optimized for efficiency; while the prediction using selected SVM classifier is very efficient. Our MKB tracking runs on a PC with 3.4-GHz CPU and 12-G memory, and takes less than 0.6-s process a frame.[3]

B. Experimental Results

We present representative tracking results in this section. More results and high resolution videos can be found at http://www.youtube.com/user/mkbtrack.

1) *Occlusion:* Fig. 4(a) shows tracking results of one sequence from the CAVIAR dataset. Our MKB tracking method is able to keep track of the target even when it is occluded by another object with similar appearance. All the other methods drift away when occlusion occurs (the OAB, MIL, SLSA, and MTT methods) or when there is a large scale change (the PF tracking method). While the FragTrack method is able to locate the target object in some frames, it does not deal with scale well as the representation does not adapt to scale or appearance change. Since we use multiple kernels to map the image data to a discriminative higher dimensional space, we have a strong classifier that is able to locate the object by separating the foreground target from the background. Table I shows the average tracking errors in terms of object center.

2) *Scale Change:* Fig. 4(b) shows the results of a sequence [19] in which the scale of the target object changes significantly. Since the frame rate of this sequence is low, the scale of the target changes drastically in a very short time. Our algorithm is able to accurately locate the target position throughout the sequence. In contrast, the OAB, FragTrack, and MIL methods lose track of the object when large scale change occurs as the adopted models do not account for scale change. Although this problem can be alleviated by extending these methods with the similarity transform, it is not clear whether these approaches are able to efficiently deal with large scale change or not. On the other hand, the PF, FragTrack, SLSA, and MTT methods do not perform well in this sequence.

3) *Illumination Change:* We evaluate whether trackers are able to handle illumination change. As shown in Fig. 4(b), there is significant illumination change in frame #30. Our MKB tracking algorithm successfully keeps track of the target even the target undergoes large illumination and scale change. However, other tracking methods except the VTD method fail to locate the target when illumination changes. Fig. 4(c) shows another sequence with illumination change. When the target object goes underneath the overpass, the PF, FragTrack, and OAB tracking methods drift away because they are not designed to deal with large illumination change. The MIL tracking method also gradually fails when the target object leaves the overpass region. The MTT method does not handle the scale change well so the tracking windows gradually drifts away. In contrast, only our MKB tracking method and SLSA method keep track of the target with high accuracy throughout the entire sequence.

4) *Distortion:* We evaluate the effect of image distortion on object tracking. Fig. 4(d) shows a surveillance video where images are acquired with a fisheye lens. The geometric shape and the aspect ratio of the objects are distorted significantly. Without geometric restoration, our algorithm is able to track the target object throughout the sequence, while all the other methods drift away or cannot accurately locate the target.

5) *Complex Background:* Fig. 4(f) shows tracking results on a sequence, in which a figure skater performs on an ice rink where variegated light changes significantly. In addition, the dark clothes of the skater are similar to the background. The MIL, VTD, OAB, and SLSA methods do not track the skater well, especially when there is large pose change in the dark background. The PF, FragTrack, and MTT algorithms lose track of the skater when the twisting motion occurs. In contrast, our algorithm is able to locate the skater well.

6) *Pose Change:* The player of interest in Fig. 4(g) exhibits multiple poses, including running, jumping, turning around, and standing. In addition, there is also illumination change in several frames. From Fig. 4(g), we observe that our method is able to track the player in different poses with comparable performance of the VTD method. The proposed tracker does not lose the target player even when illumination changes significantly (frame #710). This can be attributed to the usage of the HoG features, which are less sensitive to the lighting change. The PF, FragTrack, and OAB methods do not track the player well when he suddenly moves in different directions, as well as poses except in some frames. The SLSA and MTT methods easily fail at the beginning of the sequence when there is partial occlusion. Another example is shown in Fig. 4(e) where the head pose of Tom continuously changes. In addition, there are also occlusion and large movement. Although most methods keep track of the target object, the tracking accuracy of our MKB tracker is better than the others.

7) *Fast Motion:* Fig. 4(h) shows the tracking results of the *bolt* sequence where images are acquired from multiple camera

[3]The OAB, MIL, and Frag trackers (implemented in C or C++ languages) run faster than our MKB tracker. However, the VTD method is slower than our MKB tracker although it is also implemented by C. The PF method implemented by MATLAB runs slightly faster since it only uses color features and does not have an update stage.

TABLE I
AVERAGE TRACKING ERROR (IN TERMS OF CENTER POSITION). THE SYMBOL ''-'' MEANS THAT A METHOD LOSES TRACK OF THE OBJECT AT THE BEGINNING OF THE SEQUENCE

Sequence	OAB	MIL	PF	Frag	VTD	SLSA	MTT	MKB	SVM1	SVM2	AC-	MKT	LMKT
Caviar 1	67.6	68.3	29.4	10.0	5.7	62.0	65.0	**4.5**	4.8	19.8	**2.8**	10.4	7.9
singer	49.5	73.8	137.6	-	**10.1**	38.3	24.8	**19.2**	46.6	54.2	24.7	50.6	64.0
car	11.7	28.3	155.4	97.8	**5.9**	**2.6**	17.5	6.7	86.7	90.6	97.9	116.0	94.4
Caviar 2	13.6	8.4	43.5	18.2	49.9	33.6	35.1	**5.4**	152.4	108.4	**7.5**	40.4	133.5
tom1	22.1	21.0	43.9	28.2	**18.3**	81.3	60.0	**15.5**	83.4	122.5	31.4	49.2	49.9
skate	20.8	14.0	37.2	25.0	19.1	15.9	29.0	**7.3**	26.3	23.4	**13.3**	33.8	14.9
basketball	70.0	76.8	85.4	84.6	**4.4**	268.1	236.1	**8.8**	93.2	101.2	67.5	93.1	43.2
bolt	-	-	**42.1**	121.6	-	-	-	**7.6**	80.9	94.5	83.3	90.8	-
animal	68.0	66.5	169.9	-	12.0	**8.0**	**9.2**	24.5	89.3	95.3	277.4	94.1	90.6
bird2	19.2	**18.5**	42.3	29.7	54.7	57.6	63.5	20.5	93.6	113.9	**18.8**	119.7	112.8

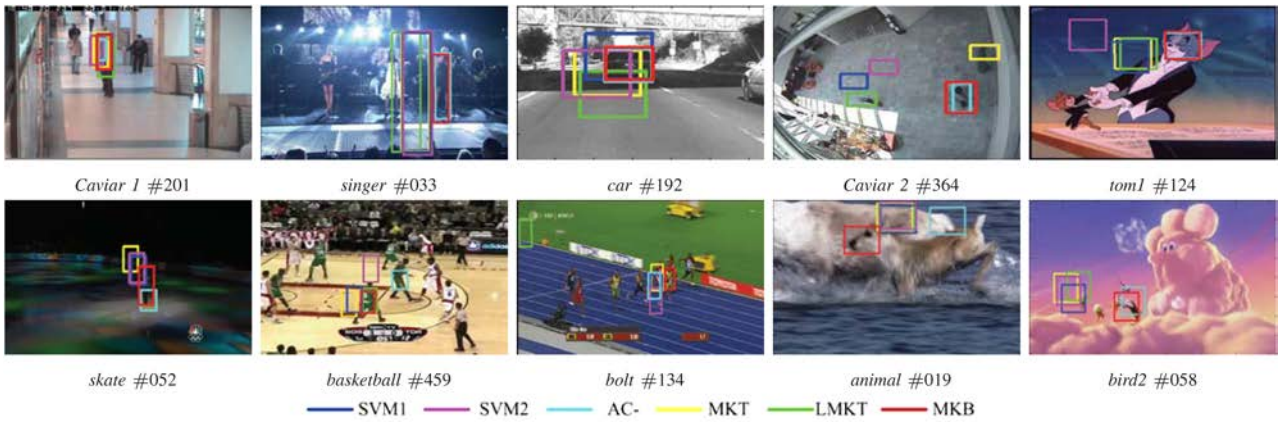


Fig. 5. More tracking results of the methods using SVM classifier with linear kernel and HoG features (blue), SVM classifier with sigmoid kernel and SIFT features (magenta), MKB tracking without affinity constraints (cyan), the standard MKL algorithm for tracking (yellow), the localized MKL algorithm for tracking (green), and the proposed MKB algorithm with affinity constraints (red).

views. The MIL, VTD, OAB, SLSA, and MTT methods do not locate the object well and drift away in just a few frames when the runner begins to accelerate. Although designed for handling various motions, the VTD method does not track the runner well due to large pose change. The FragTrack method is able to keep track of the runner until frame #141. While the PF algorithm performs well in some frames, the tracking results are not accurate. Our MKB tracking method accurately locates the runner throughout the entire sequence with favorable results against the state-of-the-art approaches. Fig. 4(i) shows another example of fast motion. All tracking methods except the VTD, MKB, SLSA, and MTT methods fail at the beginning of the sequence as they drift away from the target objects. The MKB tracking algorithm is able to deal with fast object motion and achieves comparable performance as the MTT method.

8) *Nonrigid Object:* Our MKB tracking method is also able to track nonrigid objects, as shown in Fig. 4(j). Since the target undergoes nonrigid shape deformation, the bounding box of the tracker contains a significant number of pixels from the background. The MTT method loses the target object at the beginning of the sequence. When the target object is occluded in frame #16, the PF, OAB, VTD, and SLSA tracking methods drift away. Although the FragTrack method is able to handle simple occlusion well, the tracking accuracy is not as good as the proposed algorithm.

C. Discussion

We discuss some properties of the proposed MKB algorithm with comparisons to related methods. Table I shows quantitative evaluation of the tracking algorithms on ten sequences where the average errors in terms of central tracked position with respect to the ground truth are presented.

1) *Quantitative Evaluation:* In addition to the qualitative comparisons presented in the previous section, we show quantitative comparisons in Fig. 6 and Table I. These results show that our tracker consistently performs favorably against the state-of-the-art algorithms. In addition, the error plots in Fig. 6 demonstrate that the tracking errors of our algorithm do not fluctuate throughout the sequences with lower values than the other methods.

2) *MKB Tracking Versus SVM Tracking:* We compare the tracking results using the proposed MKB algorithm and the standard SVM classifier (i.e., with one kernel and one type of feature).

We observe that empirically, SVM classifiers using a linear kernel with HoG features (referred to as SVM1), and a sigmod kernel with SIFT features (referred to as SVM2) perform well. Thus, we compare the proposed MKB method with these two types of SVMs. The results in Table I demonstrate the merits of the proposed algorithm using multiple combinations of kernels and features. Both the average position errors of the two methods are much larger than those of our MKB

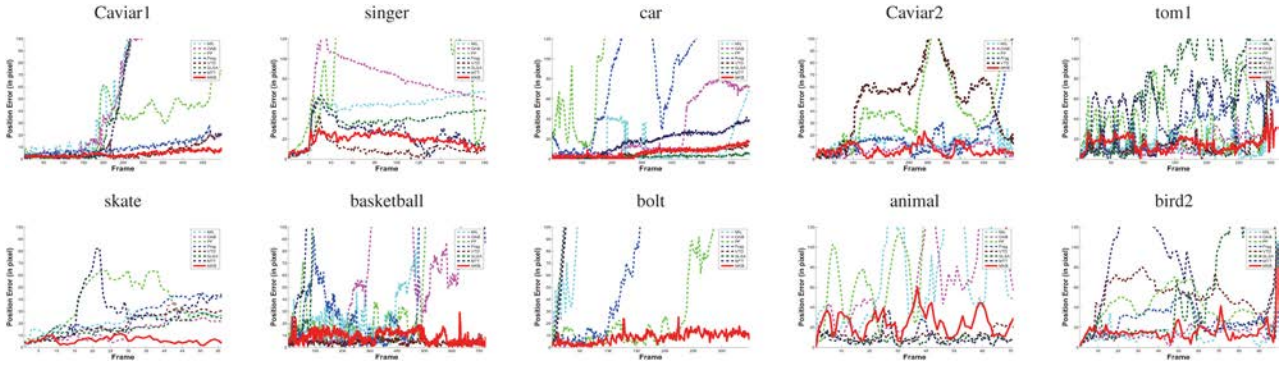


Fig. 6. Error plots of some state-of-the-art approaches and ours.

tracking method; although in some cases the methods with one single kernel perform reasonably well against other state-of-the-art approaches (e.g., SVM1 in the *Caviar 1* and *singer* sequences). Nevertheless, the performance of these two single kernel tracking methods fluctuates among image sequences. Fig. 5 shows tracking results of these tracking algorithms against the proposed MKB algorithm. The results show that our MKB tracker achieves more stable tracking performance in the presence of occlusion, large scale change, complex background, and illumination variation. The single kernel tracking methods easily drift away even at the beginning of the evaluated image sequences. With multiple kernels and features, our MKB tracker is equipped with more functions to better account for appearance change.

3) *Affinity Constraints:* To quantitatively analyze of the effectiveness of the affinity constraints in our tracking algorithm, we evaluate the tracking results without such constraints (see the "AC-" column in Table I). The results show that the use of affinity constraints facilities the MKB algorithm to achieve lower errors in almost all the cases. Fig. 5 also demonstrates the merits of these affinity constraints.

4) *MKB Tracking and MKL Tracking:* Since the proposed MKB algorithm is related to the MKL approaches, we apply MKL to tracking for evaluations (referred to as the MKT method). In addition, we evaluate a tracking method using the MKL method with localized kernels [32] (referred to as the LMKT method). We use the same settings in these methods, i.e., same set of kernels and features as well as parameters.

We observe that the training time of the MKT and LMKT methods is significantly longer than that of our method. In contrast, our MKB tracking method does not require iterations, thus more efficient in training. On average, the training time of our MKB tracking method is less than 30% of training time of the MKT or LMKT methods. The average execution time of our method for each frame is only half of that of the MKT or LMKT methods. As shown in Fig. 5, the tracking results of the MKT and LMKT methods are much worse than those of our MKB tracking method. Table I also shows that both the MKT and LMKT methods perform poorly with larger errors, and worse than the MKB tracking method without affinity constraints. The experimental results demonstrate that a straightforward application of MKL or LMKL to object tracking does not perform well. In contrast, the proposed MKB algorithm is able to achieve robust performance efficiently. The reason is that both MKL and LMKL algorithms require a large number of training samples (which are usually available in object categorization) for good performance. However, the number of samples for visual tracking are limited which may lead to reliable results, so the tracker drifts away gradually. In the MKB algorithm, we first choose a small number of reliable kernels that fit the training set well at the beginning and then refine them by using newly tracked results. Without the global optimization over the entire kernel space, our algorithm is more flexible.

VII. Conclusion

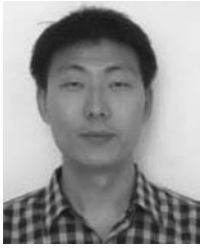
In this paper, we propose an efficient multiple kernel boosting algorithm for robust tracking. For a binary classification task, we construct a group of SVM classifiers where each one is constructed with different kernel functions and features. We treat individual single kernel SVM classifiers as weak classifiers and utilize boosting technique to adaptively select some of them to form a final decision function. Different from ensemble learning, the MKB algorithm emphasizes on the effective selection of multiple combinations of features and kernels. To further improve the discriminative ability of the strong classifier formed in the MKB algorithm, we apply affinity constraints to each selected SVM classifier. An update scheme to reselect good SVM classifiers, adjust the weights and compute affinity constraints is also included. Experiments on challenging sequences show that our MKB tracking algorithm performs favorably against the state-of-the-art methods in handling occlusion, scale change, illumination change, fast motion, and complex background.

Our future work will focus on other kernel functions and features to improve the tracking performance, e.g., the intersection kernel that is suitable for histogram comparison. Furthermore, in constructing the affinity constraint we have a basic assumption that the distribution of the support vectors can be modeled by a Gaussian distribution. Thus, more sophisticated approaches able to deal with more complex cases need to be devised. We will also optimize our implementation to reduce the processing time so that it can be applied to real-time visual tracking.

References

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

[2] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," in *Proc. ECCV*, 1996, pp. 329–342.
[3] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. CVPR*, 2000, pp. 2142–2150.
[4] D. A. Ross, J. Lim, and M.-H. Yang, "Adaptive probabilistic visual tracking with incremental subspace update," in *Proc. ECCV*, 2004, pp. 470–482.
[5] H. Wang, D. Suter, K. Schindler, and C. Shen, "Adaptive object tracking based on an effective appearance filter," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 9, pp. 1661–1667, Sep. 2007.
[6] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Int. J. Comput. Vision*, vol. 77, nos. 1–3, pp. 125–141, 2008.
[7] D. Wang, H. Lu, and M.-H. Yang, "Least soft-threshold squares tracking," in *Proc. CVPR*, 2013, pp. 2371–2378.
[8] M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density," in *Proc. ECCV*, 1996, pp. 343–356.
[9] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Proc. ECCV*, 2002, pp. 661–675.
[10] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, Aug. 2004.
[11] S. Avidan, "Ensemble tracking," in *Proc. CVPR*, 2005, pp. 494–501.
[12] 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.
[13] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. CVPR*, 2006, pp. 260–267.
[14] H. Wang and D. Suter, "A consensus-based method for tracking: Modelling background scenario and foreground appearance," *Pattern Recognit.*, vol. 40, no. 3, pp. 1091–1105, 2007.
[15] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Proc. CVPR*, 2009, pp. 983–990.
[16] X. Li, A. Dick, H. Wang, C. Shen, and A. van den Hengel, "Graph mode-based contextual kernels for robust SVM tracking," in *Proc. ICCV*, 2011, pp. 1156–1163.
[17] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," in *Proc. ICCV*, 2011, pp. 1323–1330.
[18] C. Shen, J. Kim, and H. Wang, "Generalized kernel-based visual tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 1, pp. 119–130, Jan. 2010.
[19] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Proc. CVPR*, 2010, pp. 1269–1276.
[20] X. Li, A. Dick, C. Shen, A. van den Hengel, and H. Wang, "Incremental learning of 3D-DCT compact representations for robust visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 4, pp. 863–881, Apr. 2013.
[21] X. Mei and H. Ling, "Robust visual tracking using L1 minimization," in *Proc. ICCV*, 2009, pp. 1436–1443.
[22] D. Wang, H. Lu, and M.-H. Yang, "Online object tracking with sparse prototypes," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 314–325, Jan. 2013.
[23] B. Liu, J. Huang, L. Yang, and C. A. Kulikowski, "Robust tracking using local sparse appearance model and k-selection," in *Proc. CVPR*, 2011, pp. 1313–1320.
[24] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Proc. CVPR*, 2012, pp. 1838–1845.
[25] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proc. CVPR*, 2012, pp. 1822–1829.
[26] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *Proc. CVPR*, 2012, pp. 2042–2049.
[27] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *Proc. ICCV*, 2009, pp. 221–228.
[28] B. Siddiquie, S. Vitaladevuni, and L. Davis, "Combining multiple kernels for efficient image classification," in *Proc. WACV*, 2009, pp. 1–8.
[29] F. Yang, H. Lu, and Y.-W. Chen, "Human tracking by multiple kernel boosting with locality affinity constraints," in *Proc. ACCV*, 2010, pp. 39–50.
[30] F. Bach, G. Lanckriet, and M. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. ICML*, 2004, pp. 6–13.
[31] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "Simplemkl," *J. Mach. Learning Res.*, vol. 9, no. 1, pp. 2491–2521, 2008.
[32] M. Gönen and E. Alpaydin, "Localized multiple kernel learning," in *Proc. ICML*, 2008, pp. 352–359.
[33] M. Christoudias, R. Urtasun, and T. Darrell, "Bayesian localized multiple kernel learning," EECS Dept., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-96, Jul. 2009.
[34] L. Cao, J. Luo, F. Liang, and T. Huang, "Heterogeneous feature machines for visual recognition," in *Proc. ICCV*, 2009, pp. 1095–1102.
[35] J. Yang, Y. Li, Y. Tian, L. Duan, and W. Gao, "Group-sensitive multiple kernel learning for object categorization," in *Proc. ICCV*, 2009, pp. 436–443.
[36] A. Blum and T. M. Mitchell, "Combining labeled and unlabeled SATA with co-training," in *Proc. COLT*, 1998, pp. 92–100.
[37] S. P. Abney, "Bootstrapping," in *Proc. ACL*, 2002, pp. 360–367.
[38] V. Sindhwani, P. Niyogi, and M. Belkin, "Beyond the point cloud: From transductive to semi-supervised learning," in *Proc. ICML*, 2005, pp. 824–831.
[39] I. Muslea, S. Minton, and C. A. Knoblock, "Active learning with multiple views," *J. Artif. Intell. Res.*, vol. 27, no. 1, pp. 203–233, 2006.
[40] B. Long, P. S. Yu, and Z. M. Zhang, "A general model for multiple view unsupervised learning," in *Proc. SDM*, 2008, pp. 822–833.
[41] F. Tang, S. Brennan, Q. Zhao, and H. Tao, "Co-tracking using semi-supervised support vector machines," in *Proc. ICCV*, 2007, pp. 1–8.
[42] C. Christoudias, R. Urtasun, and T. Darrell, "Multi-view learning in the presence of view disagreement," in *Proc. UAI*, 2008, pp. 88–96.
[43] Z. Yin, F. Porikli, and R. T. Collins, "Likelihood map fusion for visual object tracking." in *Proc. WACV*, 2008, pp. 1–7.
[44] M. Yang, F. Lv, W. Xu, and Y. Gong, "Detection driven adaptive multi-cue integration for multiple human tracking," in *Proc. ICCV*, 2009, pp. 1554–1561.
[45] H. Lu, S. Lu, D. Wang, S. Wang, and H. Leung, "Pixel-wise spatial pyramid-based hybrid tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 9, pp. 1365–1376, Sep. 2012.
[46] D. Wang, H. Lu, and Y.-W. Chen, "Object tracking by multi-cues spatial pyramid matching." in *Proc. ICIP*, 2010, pp. 3957–3960.
[47] J. H. Yoon, D. Y. Kim, and K.-J. Yoon, "Visual tracking via adaptive tracker selection with multiple features," in *Proc. ECCV*, 2012, pp. 28–41.
[48] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. CVPR*, 2005, pp. 886–893.
[49] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. ICCV*, 1999, pp. 1150–1157.
[50] M. Gönen and E. Alpaydin, "Localized algorithms for multiple kernel learning," *Pattern Recognit.*, vol. 46, no. 3, pp. 795–807, 2013.
[51] J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.
[52] S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with Gaussian kernel," *Neural Comput.*, vol. 15, no. 7, pp. 1667–1689, 2003.
[53] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Proc. CVPR*, 2006, pp. 798–805.
[54] A. Vedaldi and B. Fulkerson. (2008). *VLFeat: An Open and Portable Library of Computer Vision Algorithms* [Online]. Available: http://www.vlfeat.org/



Fan Yang (S'10) received the M.Sc. degree in signal and information processing from Dalian University of Technology, Dalian, China, in 2011. He is currently pursuing the Ph.D. degree at the Department of Computer Science, University of Maryland, College Park, MD, USA.

He has published several papers in visual tracking. His research interests include computer vision and pattern recognition.

Mr. Yang was a recipient of the Dean's Fellowship in 2011 and 2012.

Huchuan Lu (SM'12) received the M.Sc. degree in signal and information processing and the Ph.D. degree in system engineering from Dalian University of Technology (DUT), Dalian, China, in 1998 and 2008, respectively.

He has been a Faculty Member since 1998 and a Professor since 2012 with the School of Information and Communication Engineering, Faculty of Electronic Information and Electrical Engineering, DUT. In recent years, he has focused on visual tracking and segmentation. His research interests include computer vision and pattern recognition.

Dr. Lu serves as an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: PART B.

Ming-Hsuan Yang (SM'06) received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2000.

He is currently an Assistant Professor of electrical engineering and computer science with the Department of Electrical Engineering and Computer Science, University of California, Merced (UC Merced), CA, USA. He was a Senior Research Scientist with the Honda Research Institute, focusing on vision problems related to humanoid robots. He coauthored the book, *Face Detection and Gesture Recognition for Human-Computer Interaction* (Kluwer Academic, 2001).

Dr. Yang was a recipient of the Ray Ozzie Fellowship for his research work in 1999. He edited a special issue on face recognition for *Computer Vision and Image Understanding* in 2003 and a special issue on real world face recognition for the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He served as an Area Chair for the IEEE International Conference on Computer Vision in 2011, the IEEE Conference on Computer Vision and Pattern Recognition in 2008 and 2009, and the Asian Conference on Computer in 2009, 2010, and 2012. He served as an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE from 2007 to 2011 and was an Associate Editor of the *Image and Vision Computing*. He was a recipient of the NSF CAREER Award in 2012, the Campus-Wide Senate Award for Distinguished Early Career Research at UC Merced in 2011, and the Google Faculty Award in 2009. He is a Senior Member of the ACM.