Human Tracking by Multiple Kernel Boosting with Locality Affinity Constraints

Fan Yang[1], Huchuan Lu[1], and Yen-Wei Chen[1,2]

[1]School of Information and Communication Engineering,
Dalian University of Technology, Dalian, China
[2]College of Information Science and Engineering,
Ritsumeikan University, Kusatsu, Japan

Abstract. In this paper, we incorporate the concept of Multiple Kernel Learning (MKL) algorithm, which is used in object categorization, into human tracking field. For efficiency, we devise an algorithm called Multiple Kernel Boosting (MKB), instead of directly adopting MKL. MKB aims to find an optimal combination of many single kernel SVMs focusing on different features and kernels by boosting technique. Besides, we apply Locality Affinity Constraints (LAC) to each selected SVM. LAC is computed from the distribution of support vectors of respective SVM, recording the underlying locality of training data. An update scheme to reselect good SVMs, adjust their weights and recalculate LAC is also included. Experiments on standard and our own testing sequences show that our MKB tracking outperforms some other state-of-the-art algorithms in handling various conditions.

1 Introduction

Visual tracking has been popular in the computer vision community for decades. In this paper, we consider tracking as a binary classification, aiming to discriminate the object from the background in successive frames. Collins *et al.* [1] propose a method to adaptively select color features that best separate the object from the background. Grabner *et al.* [2] design an online version of Adaboost classifier for object tracking, which accumulates samples to train a strong classifier and then use the classifier to find the object in videos. To solve drifting problem, SemiBoost tracker [3], also a boosting classifier combined with semi-supervised learning, is proposed. Avidan proposes support vector tracking (SVT) [4] which utilizes an off-line SVM to discriminate the target vehicle from the background, and an ensemble tracking approach [5]. The main concept of "ensemble" is to collect a number of weak classifiers to learn the difference between the object and the background, and then iteratively train new weak classifiers to replace old ones. Tian *et al.* [6] devise an ensemble SVM classifier based tracking algorithm. They use linear SVM to automatically select "key frame" of the target as support vectors. By combining several linear SVM classifiers, history information is integrated into the tracking framework. More recently, Babenko *et al.* [7] propose

a tracking framework utilizing Multiple Instance Learning (MIL) algorithm to augment training and update samples.

Noticing that the SVM-based classifier can effectively solve classification problem in tracking field, we focus on the kernel learning technique used in object classification. The basic idea of kernel used in non-linear SVM is to map training samples from the input space to a higher dimensional feature space, where they are linearly separable, without explicitly defining the mapping function. In particular, we are interested in Multiple Kernel Learning (MKL) [8–10], which has shown great advantages in the recent object classification task [11, 12]. MKL aims to learn an optimal kernel combination and assign appropriate weight to each kernel in supervised learning settings. Standard MKL displays remarkable ability to solve multi-class classification problems. However, for better classification, many improvements have been proposed. Rakotomamonjy *et al.* [8] propose an improved MKL algorithm, named SimpleMKL, for simplifying the optimization process based on mixed-norm regularization. Localized MKL (LMKL) [13] and Bayesian Localized MKL (BLMKL) [14] are devised to exploit the distribution of training data on each kernel space and give higher weights to appropriate kernel functions if data has underlying localities. Motivated by LMKL, Cao *et al.* [15] propose Heterogeneous Feature Machines (HFM) to learn a non-linear combination of multiple kernels; Yang *et al.* [16] propose group-sensitive multiple kernel learning (GS-MKL) to accommodate the intra-class diversity and the inter-class correlation for object categorization. Boosting method is also incorporated into MKL to implement feature combination [17] and feature selection [18].

Impressed by the remarkable performance of MKL, we propose a Multiple Kernel Boosting (MKB) algorithm with Locality Affinity Constraints (LAC) for human tracking. To describe an object, we use 3 feature descriptors, RGB histogram, Histogram of Gradient (HoG) [19] and SIFT [20]; to map the input space to the kernel space, we use 4 kernels, linear kernel, polynomial kernel, RBF kernel and sigmoid kernel. We consider each single kernel SVM as a "weak classifier". To find the best combination of these SVMs, we utilize boosting technique instead of a global optimization used in most MKL algorithms. We also introduce locality affinity information of input data, which is computed from the distribution of support vectors of the respective single kernel SVM, into the final decision function. In each new frame, we apply particle sampling to generate a number of candidates. Tracking is then accomplished by finding the best candidate. For update, we retrain the set of single kernel SVMs, reselect some discriminative ones by MKB, and recalculate LAC.

The remainder of the paper is organized as follows: Section 2 and Section 3 introduce our Multiple Kernel Boosting (MKB) algorithm and Locality Affinity Constraints (LAC) respectively. Main tracking framework is in Section 4 and experimental results on various sequences are shown and discussed in Section 5. The last section gives out conclusion.

2 Multiple Kernel Boosting

2.1 Standard MKL

The main difficulty of single SVM is to choose a proper kernel for the given training dataset. However, MKL aims to find an optimal convex combination of multiple kernels and the associated classifier simultaneously. For binary classification, assuming that we have training samples $\{x_i, y_i\}_{i=1}^{D}$, where x_i is the i^{th} sample and $y_i = \{\pm 1\}$ indicates the label of the sample, our task is to train a multi-kernel based classifier $F(x)$ to classify an unlabeled sample into a class. Let $\{K_m\}_{m=1}^{M}$ be the kernel matrices computed for different feature modalities. 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 $\sum_{m=1}^{M} \beta_m = 1$. K_m can be the same kernels with different hyperparameters or different kernels. Also, they can be applied to different feature sets. 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 $\{\alpha_i\}$ and b are the Lagrange multipliers and the bias in the standard SVM algorithm. We can learn $\{\alpha_i\}$, $\{\beta_m\}$ and b from a joint optimization process. Details can be found in [10].

2.2 Multiple Kernel Boosting

Despite its success in object categorization, MKL cannot be directly applied to tracking due to time-consuming optimization process, large amount of training samples and constant weights. However, Gehler and Nowozin [17] have discussed a boosting version of MKL for feature combination, which inspires us to propose Multiple Kernel Boosting (MKB) for tracking applications. For a sample x, we construct a vector by concatenating its kernel values with all the training samples $\{x_i, y_i\}_{i=1}^{D}$ to indicate the m^{th} kernel response

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

So we can rewrite Equation 2 as the following form

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

where $\alpha = (\alpha_1 y_1, \alpha_2 y_2, ..., \alpha_D y_D)^T$. So we convert standard MKL to a linear combination of the real value output of M separate SVMs $K_m(x)^T\alpha + b$. According to [17], we can separately train M SVMs with different parameters $\{\alpha_m, b_m\}$ at first, and then optimize $\{\beta_m\}$ in the second step. Each individual SVM is not restricted to share the same parameter. By letting $h_m(x) = K_m(x)^T\alpha + b$, we convert the decision function of standard MKL to $F(x) = \sum_{m=1}^{M} \beta_m h_m(x)$. To determine $\{\beta_m\}$, we can simply use other methods. In this paper, we use boosting method, so we name our algorithm Multiple Kernel Boosting (MKB). In the boosting form, the decision function can be written as

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

where L indicates the iteration time. We regard MKL as choosing multiple "weak" single kernel SVMs into a final strong classifier. MKB avoids complex global optimization, thereby making the concept of MKL applicable to tracking.

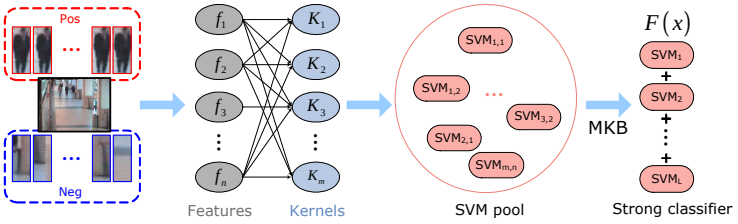

Fig. 1. Illustration of Multiple Kernel Boosting (MKB) process.

As Figure 1 shows, we extract $\{f_1, f_2, ..., f_N\}$ features from a set of positive and negative samples and send them into $\{K_1, K_2, ..., K_M\}$ kernels. Then we get $M \times N$ combinations; for each combination, we train a single kernel SVM. The classification error of a single kernel SVM 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 $x > 0$, otherwise it equals 0. $w(i)$ is training samples' weight. $h(x_i)$ is the real value classification output of the SVM on the input x_i. We aim to adaptively select multiple features and kernels that are of the most discriminative ability from the pool. So we use boosting technique to iteratively choose an SVM and add it to the final decision function. The complete process of MKB is shown in Algorithm 1.

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 samples' weights $w_1(i) = 1/D$

3: for $l = 1$ to L do

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

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

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 $F(x) \leftarrow F(x) + \beta_l h_l(x)$

5) $w_{l+1}(i) = \frac{w_l(i)}{Z_l} e^{-\beta_l y_i h_l(x_i)}$

4: end for

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

3 Locality Affinity Constraints

Although MKB produces promising tracking results, we find that it is not stable enough in some cases. So we try to improve the original MKB. Motivated by LMKL [13] and GS-MKL [16], we incorporate the distribution of training data into $F(x)$ to enhance the robustness of MKB. Rewriting Equation 2, we obtain

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

where $\beta_m(x)$ is a function of input x, rather than a constant β_m in the standard MKL. It can be learned from an iteration algorithm [13]. However, we find that the optimization process is intolerantly time-consuming [16]. Moreover, iteration cannot guarantee convergence to global optimum and unsuitable initial parameters may also degrade the performance. Considering the problem of limited training samples in tracking, we devise a simple but effective method to exploit the underlying distribution of training data.

We assume that an SVM trained in MKB has recorded the property of training data with respect to the feature and kernel. Since support vectors of each SVM reserve most information, we utilize those support vectors for computing the locality of data. Letting $\beta_l = \beta_l^* A_l(x)$ and rewriting Equation 5, we get

$$F(x) = \sum_{l=1}^{L} \beta_l^* A_l(x) h_l(x) \tag{8}$$

where β_l^* is the same as β_l in Equation 5, which is calculated by MKB. $A_l(x)$ is a function of input x, indicating the similarity of x with the trained SVM,

which is called Locality Affinity Constraint (LAC) in our algorithm. Locality affinity means that if the input sample complies with the distribution of support vectors in a specific SVM, we think that the importance of the corresponding SVM is high, thus assigning it larger weight. We construct a probability model to describe the locality affinity, which is defined as

$$A_l(x) = 1 - \exp(-|\sigma_l(x)|) \tag{9}$$

where $\sigma_l(x) = \log\left[\frac{p_l(y=1|x)}{p_l(y=-1|x)}\right]$. For each trained SVM $h_l(x)$, we compute the mean μ_l^+ and μ_l^- of positive and negative support vectors respectively. Then $p_l(y=1|x)$ and $p_l(y=-1|x)$ are computed as follows

$$p_l(y^*|x) = \exp(-|x - \mu_l^*|) \tag{10}$$

where $y^* = 1$ or $y^* = -1$ when μ_l^* is μ_l^+ or μ_l^-. Here, $A_l(x) \in (0, 1)$, which can be seen as the probability of sample x belonging to the support vectors. If x is similar with training data on a specific combination of feature and kernel, the importance of the corresponding SVM is high, and vice versa. Therefore, we formulate the distribution of training samples and impose such constraints on testing samples, thereby improving the discriminative ability of the decision function.

4 Main Tracking Framework

In this section, we will introduce how tracking proceeds based on the aforementioned algorithms. In 1st frame, we draw a bounding box x^1 enclosing the object we want to track, where $x^1 = (c_x^1, c_y^1, s^1, \theta^1)$ records the center, size and rotation angle of the object. The superscript indicates the current frame number. To augment the number of training samples, we crop out a set of images $X^+ = \{x_i | 0 \leq l(x_i) - l(x^1) < r_\alpha\}_{i=1}^{D^+}$ to collect positive samples. Here r_α is a small constant and $l(x)$ indicates the center of x. Similarly, we crop out a set of negative samples $X^- = \{x_i | r_\beta \leq l(x_i) - l(x^1) < r_\gamma\}_{i=1}^{D^-}$. We set $r_\beta > r_\alpha$ to allow less than 1/4 overlap between positive and negative samples. Note that we only use 1st frame to collect $(D^+ + D^-)$ training samples. Extracting features on these samples, performing MKB and adding locality affinity functions, we obtain a multi-kernel based decision function.

To improve efficiency, we adopt particle sampling technique in the following frames. The predicting distribution of x^t given all available observations $z^{1:t-1} = \{z^1, z^2, ..., z^{t-1}\}$, denoted by $p(x^t|z^{1:t-1})$, is recursively computed as

$$p(x^t|z^{1:t-1}) = \int p(x^t|x^{t-1}) p(x^{t-1}|z^{1:t-1}) dx^{t-1} \tag{11}$$

When the observation z^t is obtained at time t, the state vector is updated as

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

Algorithm 2 MKB Tracking with Locality Affinity Constraints

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 $F(x) = 0$, empty sample queue Q
Output: tracking results in each frame $\{x^1, x^2, ..., x^t\}$
For the first frame I_t $(t = 1)$
1: given the bounding box $x^1 = (c_x^1, c_y^1, s^1, \theta^1)$, extract D^+ positive samples and D^- negative samples
2: extract features $\{f_n(x_i)\}_{i=1}^{D^+ + D^-}$ and train individual single kernel SVMs $h_{m,n}(x)$
3: compute the locality affinity function $A_{m,n}(x)$ according to the distribution of support vectors for each trained SVM $h_{m,n}(x)$
4. apply **Algorithm 1** to obtain the strong classifier $F(x) = \sum_{l=1}^{L} \beta_l^* A_l(x) h_l(x)$
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 compute 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: the tracked object is find by $x^t = \sum_{i=1}^{D} w_i^t x_i^t$
4: regard x^t as positive sample and collect 4 negative samples around x^t, push them into the sample queue Q
5: if the length of sample queue $Length(Q) = 5T_u$, do
  1) select SVM $h_{m,n}(x)$ from weak SVM pool h, extract feature $S_Q = f_n(x), x \in Q$. Form new training sample groups $S'_{m,n} = S_{m,n} \cup S_Q$, where $S_{m,n}$ are support vectors of $h_{m,n}(x)$. Train $h_{m,n}(x)$ again using $S'_{m,n}$
  2) remove $h_{m,n}(x)$ from the pool h
  3) repeat 1) and 2) until the pool is empty
  4) update $\mu_{m,n}^+$ and $\mu_{m,n}^-$ of new trained support vectors to obtain new $A_{m,n}(x)$
  5) perform **Algorithm 1** again to reselect appropriate $h_l(x)$ to form a new $F(x) = \sum_{l=1}^{L} \beta_l^* A_l(x) h_l(x)$
  6) clean up the sample queue Q
6: otherwise output x^t and proceed to the next frame

where $p(z^t|x^t)$ 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 $q(x^t|x^{1:t-1}, z^{1:t})$. We let $q(x^t|x^{1:t-1}, z^{1:t}) = p(x^t|x^{t-1})$ then the weights $w_i^t = w_i^{t-1} p(z^t|x_i^t)$. We think that $p(x^t|x^{t-1})$ complies with a Gaussian distribution and affine parameters in x^t are independent. So in frame I_t, we have D candidates with different affine parameters around the tracked object x^{t-1} in frame I_{t-1}. Then we apply $p\left(z^t|x_i^t\right) = e^{F(x_i^t)}$ to compute $p(z^t|x_i^t)$, which is particle's weight. Subsequently, we normalize $\{w_i^t\}_{i=1}^{D}$ and compute the weighted sum of particles to find the object, denoted as $x^t = \sum_{i=1}^{D} w_i^t x_i^t$.

Moreover, to capture the variance of the object, we also incorporate an update scheme into the tracking framework. In each frame, we consider the tracked object x^t as positive sample, and extract four negative samples from four directions (up, down, left, right) without overlap with the positive one. We accumulate these samples for T_u frames, then retrain individual SVMs using new samples and corresponding support vectors. Subsequently, we perform MKB again to obtain a new $F(x)$. $A_l(x)$ is also recalculated from new support vectors. The complete process is shown in Algorithm 2.

5 Experiments

5.1 Experimental Settings

We implement our tracking algorithm by Matlab. In 1st frame, both positive and negative samples are 20. The pool of weak SVMs contains 12 single kernel SVMs, each of which focuses on a specific combination of 3 features (64-dim RGB histogram, 128-dim HoG and 128-dim SIFT descriptor) and 4 kernels (linear kernel, polynomial kernel, RBF kernel and sigmoid kernel). The iteration time of MKB is 10, while the number of selected SVMs varies according to different sequences. In each new frame, we sample 200 particles according to a pre-defined distribution and send them to $F(x)$ to get 200 real values. The update rate also varies according to the property of different sequences. Note that all parameters are fixed except the distribution for sampling affine parameters of sequences.

We also run other three tracking systems: Online Adaboost tracking (OAB) [2], Multiple Instance Learning tracking (MIL) [7] and color-based particle filter tracking (PF) [21]. Similar with our MKB tracking, both OAB and MIL tracking rely on a boosting technique and use new samples to change weak classifiers and corresponding weights. Also, our approach includes the particle sampling that generates a number of candidates used to approximate the current state of the object. We are going to show that the good performance of our MKB tracking is not necessarily attributed to particle sampling, so our approach outperforms PF in most sequences. In our experiments, the number of selectors in OAB remains 100; PF uses 512-dim RGB feature (8×8×8 bins) and its sampling parameters are constant on all sequences.

5.2 Results

We compare our method with OAB, MIL and PF tracking. To better display the advantages of the proposed method, we will analyze the tracking results under various situations.

Occlusion. Figure 2 shows a comparison under occlusion. The testing sequence is from CAVIAR database. Our MKB tracking continuously keeps track of the person even when he is occluded by another person in similar color. While all other methods drift away from the object when occlusion occurs (OAB and MIL) or when the object's size changes (PF). We also find that HoG plays the

Fig. 2. Comparison of tracking results when there is occlusion. Top row shows results of our approach. Results of other approaches are shown in the bottom row.

Fig. 3. Comparison of tracking results when there is scale change. Top row shows results of our approach. Results of other approaches are shown in the bottom row.

most important part when occlusion occurs. Because the color of the two persons is almost the same, color feature is unreliable.

Scale change. To test the ability to handle the object's scale change, we also compare the four algorithms on another sequence also from CAVIAR database, as shown in Figure 3. In the sequence, a person walks away from the camera, so his size becomes smaller than that in the first few frames. There is also simple occlusion by other people. Our approach can locate the person's position accurately and the tracking result is quite stable. In contrast, lacking a scheme of adaptively adjusting the size of the tracking window, both OAB and MIL lose the object when large scale change occurs. PF is even confused by the other three persons close to the real object, even though they do not occlude the object.

Complex background. We also run the four algorithms on our own testing sequences. Figure 4 shows tracking results on a sequence, in which a figure skater exhibits a set of actions in a skating rink. As the figure shows, the background is complex, including various colors. Sometimes the dark background is even the same as the skater's black clothes. MIL, OAB and PF cannot find the precise position of the skater, especially when he changes his poses under the dark background; while PF even loses the skater when he changes his skating direction. In contrast, our approach can locate the skater, resulting in more accurate results. Therefore, our MKB tracking has the ability to deal with complex background.

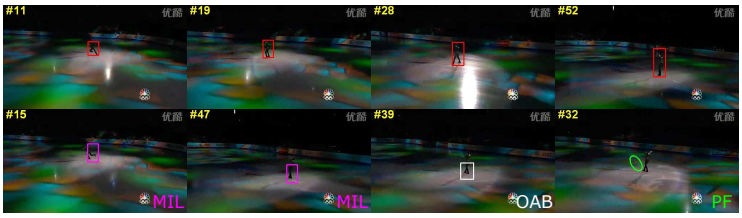

Fig. 4. Comparison of tracking results when background is complex. Top row shows results of our approach. Results of other approaches are shown in the bottom row.

Fig. 5. Comparison of tracking results when there is fast motion. Top row shows results of our approach. Results of other approaches are shown in the bottom row.

Fast motion. The last experiment we will report is to show the ability to handle fast motion of the object. We aim to keep track of the famous sprinter Bolt in the sequence that includes part of a 100m dash competition. The main difficulty is that the runner moves fast. From Figure 5, we can see that both MIL and OAB drifts away just in the first few frames when Bolt starts to accelerate; while our MKB tracking and PF find Bolt accurately until about 120th frame. However, we also find that PF is much sensitive to the sampling parameters: slight change of initial parameters may affect the performance severely. Compared with it, our method is more robust. The change of sampling parameters within an appropriate range does not decrease the accuracy.

5.3 Discussions

In this section, we will briefly discuss some properties of our proposed method. Table 1 shows quantitative comparisons on 7 testing sequences. Numbers indicate the average error of center of the object per frame on testing sequences.

MKB tracking vs. single kernel SVM. First, we compare the proposed algorithm with single kernel SVM using only one feature. We observe that in most sequences, HoG+linear kernel and SIFT+linear kernel perform well. So we compare the tracking results of the two single kernel methods. In Table 1, S1 and S2 indicate HoG+linear kernel and SIFT+linear kernel respectively. From Table 1, we can see that only using one combination of feature and kernel cannot achieve good results. Both the average position errors of the two methods are

Table 1. The average position error per frame.

	OAB	MIL	PF	MKB	S1	S2	LAC-
ShopAssistant2cor	67.6	68.2	13.1	4.5	4.8	19.8	**2.8**
ThreePastShop2cor	15.2	17.7	35.8	**3.5**	129.8	19.8	4.2
MeetWalkSplit	9.0	21.5	9.0	**8.0**	12.0	89.4	11.9
skate	20.7	13.8	25.1	**12.4**	26.3	23.4	13.3
dash	129.4	206.9	11.1	**4.1**	18.8	12.4	16.1
Browse1	13.6	8.4	36.0	7.5	152.5	108.6	**5.4**
OneLeaveShopR1cor	7.3	10.7	8.9	**4.6**	31.8	46.2	51.6

much larger than that of our MKB tracking, although in some cases HoG+linear kernel can produce more accurate results than other state-of-the-art approaches.

MKB tracking vs. other approaches. Besides qualitative comparisons in the previous section, we also give out quantitative comparisons of our proposed tracking and other algorithms. From the table, we can see that our MKB tracking is much more robust. The adaptive selection of kernels and features shows its advantage, compared with other approaches.

Impact of LAC. To test the effectiveness of LAC, we also run our tracking system without such constraints (see "LAC-" column in Table 1). LAC shows predominant advantage in most cases, leading to lower average position error, although in only two sequences it does not outperform MKB tracking without LAC. Therefore, by incorporating LAC, we boost the performance of original MKB tracking. Moreover, we can see that even we use standard MKB for tracking, the results are not inferior to other existing approaches.

6 Conclusion

In this paper, we incorporate the concept of Multiple Kernel Learning (MKL) algorithm, which is used in object categorization, into human tracking field. We devise an algorithm called Multiple Kernel Boosting (MKB), instead of directly adopting MKL. In MKB, we treat individual single kernel SVMs as weak classifiers and utilize boosting technique to adaptively select a number of good SVMs into a final decision function focusing on different features and kernels. Compared with standard MKL, MKB is much efficient. To strengthen the discriminative ability of the stong classifier formed in MKB, we also apply Locality Affinity Constraints (LAC) to each selected SVM. LAC is computed from the distribution of support vectors of respective SVM, recording the underlying locality of training data. An update scheme to reselect good SVMs, adjust their weights and recalculate LAC is also included in our tracking framework. Experiments on some standard and our own testing sequences show that our MKB tracking outperforms some of its rivals in handling simple occlusion, scale change and complex background.

Acknowledgement. The work was supported by the Fundamental Research Funds for the Central Universities, No. DUT10JS05, and the National Natural Science Foundation of China (NSFC), No.61071209.

References

1. Collins, R., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. IEEE Transactions on Pattern Analysis and Machine Intelligence (2005) 1631–1643
2. Grabner, H., Bischof, H.: On-line boosting and vision. In: CVPR. Volume 1. (2006) 260–267
3. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line boosting for robust tracking. In: ECCV. Volume 2. (2008) 234–247
4. Avidan, S.: Support vector tracking. PAMI **26(8)** (2004) 1064–1072
5. Avidan, S.: Ensemble tracking. In: CVPR. Volume 2. (2005) 494–501
6. Tian, M., Zhang, W., Liu, F.: On-line ensemble svm for robust object tracking. In: ACCV. (2007) 355–364
7. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: CVPR. (2009) 983–990
8. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: SimpleMKL. Journal of Machine Learning Research **9** (2008) 2491–2521
9. Sonnenburg, S., Ratsch, G., Schafer, C., Scholkopf, B.: Large scale multiple kernel learning. Journal of Machine Learning Research **7** (2006) 1531–1565
10. Bach, F., Lanckriet, G., Jordan, M.: Multiple kernel learning, conic duality, and the SMO algorithm. In: ICML. (2004)
11. Varma, M., Ray, D.: Learning the discriminative power-invariance trade-off. In: ICCV. (2007)
12. Kumar, A., Sminchisescu, C.: Support kernel machines for object recognition. In: ICCV. (2007)
13. Gonen, M., Alpaydin, E.: Localized multiple kernel learning. In: ICML. (2008) 352–359
14. Christoudias, M., Urtasun, R., Darrell, T.: Bayesian localized multiple kernel learning. Technical Report UCB/EECS-2009-96, EECS Department, University of California, Berkeley (2009)
15. Cao, L., Luo, J., Liang, F., Huang, T.: Heterogeneous Feature Machines for Visual Recognition. In: ICCV. (2009) 1095–1102
16. Yang, J., Li, Y., Tian, Y., Duan, L., Gao, W.: Group-Sensitive Multiple Kernel Learning for Object Categorization. In: ICCV. (2009)
17. Gehler, P., Nowozin, S.: On feature combination for multiclass object classification. In: ICCV09. (2009) 221–228
18. Siddiquie, B., Vitaladevuni, S., Davis, L.: Combining multiple kernels for efficient image classification. In: WACV09. (2009) 1–8
19. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. Volume 1. (2005) 886–893
20. Lowe, D.: Object recognition from local scale-invariant features. In: ICCV. (1999) 1150–1157
21. Nummiaro, K., Koller Meier, E., Van Gool, L.: Object tracking with an adaptive color-based particle filter. In: DAGM02. (2002) 353–360