Kernel Regression Based Online Boosting Tracking

HONGWEI HU1, BO MA1, YUWEI WU1, WEIZHANG MA1 AND KAI XIE2

1Beijing Laboratory of Intelligent Information Technology
School of Computer Science and Technology
Beijing Institute of Technology
Beijing, 100081 P.R. China

2School of Information Engineering
Beijing Institute of Graphic Communication
Beijing, 100081 P.R. China

Although online boosting algorithm has received an increasing amount of interest in visual tracking, it is susceptible to class-label noise. Slight inaccuracies in the tracker can result in incorrectly labeled examples, which degrade the classifier and cause drift. This paper proposes a kernel regression based online boosting method for robust visual tracking. A nonlinear recursive least square algorithm which performs linear regression in a high-dimensional feature space induced by a Mercer kernel is employed to derive weak classifiers. Online sparsification to filter samples in feature space is adopted to reduce the computational cost of the recursive least square algorithm. In our method, weak classifiers themselves can be modified adaptively to cope with scene changes. Experimental results compared with several relevant tracking methods demonstrate the good performance of the proposed algorithm under challenging conditions.

Keywords: kernel regression, online sparsification, Adaboost, nonlinear recursive, visual tracking

1. INTRODUCTION

Visual tracking is an important topic in computer vision with many applications, such as intelligent surveillance, human-computer interaction systems, and medical imaging, etc. The aim of tracking is to locate the targets in a video sequence. The challenges in designing a robust visual tracking algorithm are mainly caused by the presence of noise, occlusion, background clutter and illumination changes. Different tracking strategies have been implemented and applied to overcome these difficulties, from kernel-based methods [4, 13], subspace-based methods [32, 33], sparsity-based methods [30, 31], variational-based method [34] to detection-based methods [6, 22, 25]. A thorough review can be found in [1].

Recently, online boosting methods have been used successfully for tracking. Avidan [3] proposed an online method to learn an ensemble of weak classifiers for visual tracking. Subsequently, Grabner and Bischof [23] applied online boosting to object detection and visual tracking. They proposed an online feature selection method, where a group of selectors is initialized randomly. This method was later extended in a semi-supervised setting in [22] using the semi-boosting framework, where labeled examples
were obtained from the first frame and subsequent examples were incorporated in a semi-supervised manner.

Nevertheless, the above-mentioned classifiers are difficult to decide autonomously both the positive and negative samples. Moreover, they rely mostly on the recently arrived data and thus the classifiers may not be correctly trained if the data are noisy. If these errors accumulate over time, they finally lead to drifting of the tracker. Babenko et al. [6] formulated the tracking task as a multiple-instance learning (MIL) problem. However, the instances in bags are not selected effectively due to the use of Noisy-OR (NOR) model. To eliminate such instances which are less effective for classification, Zhou et al. [26] proposed an online tracking algorithm in which the support instances are selected adaptively within the multiple instance learning framework. Zeisl et al. [27] presented an online semi-supervised multiple instance boosting for tracking. This method combines the robustness of semi-supervised updates towards occlusions and the flexibility of multiple instance learning on where to select positive updates. Alternatively, a semi-supervised learning approach named P-N learning is developed by Kalal et al. [29] to train a binary classifier with structured unlabeled data.

In this paper, we introduce the adaptive nonlinear weak classifiers for online learning and tracking, in which base learners can be autonomously updated. To classify the nonlinear separable data obtained from the image sequence, we apply kernel recursive least square (KRLS) as the weak classifier in our boosting framework. During tracking, an online sparsification algorithm is performed to reduce the computational cost. The kept samples are adopted as the base samples for constructing the nonlinear weak classifiers. And the strong classifier is updated by the tracking results with a recursion form. Fig. 1 shows the block diagram of the proposed visual tracking framework. Our algorithm combines kernel methods and boosting, which is able to obtain a better tracking precision. In addition, the online sparsification is applied to solve the efficiency problem brought by kernel.

![Fig. 1. Target tracking scheme based on online boosting with kernel recursive least square.](image)

The main contribution of this paper is three-fold. (1) We introduce KRLS as the weak classifier which has excellent classification performance in the proposed tracking algorithm. (2) In order to reduce the complexity when constructing a weak classifier, the online sparsification is adopted to update the dictionary using selected samples. (3) Based on AdaBoost framework, we propose to update the tracker using KRLS based online boosting, and our tracking algorithm could obtain good performance under challenging conditions.
2. RELATED WORK

This work is associated with two main aspects: kernel methods and online learning methods. We briefly review the relevant literatures and basic theories in this section. Kernel methods have been very popular in machine learning and pattern recognition (e.g., support vector machines (SVMs) [7] and object detection [20]). The kernel methods utilize Mercer kernel function applied to pairs of input vectors to produce nonlinear versions of conventional linear supervised and unsupervised learning algorithms [8]. The basic idea behind kernel methods is that the mercer kernel function can be interpreted as an inner product in a high dimensional Hilbert space (feature space). The kernel methods applied to visual tracking can be traced back to pioneering work [4]. Comaniciu et al. [4] used histogram as target representation that is regularized by spatial masking. A metric derived from the Bhattacharyya coefficient as similarity measure is employed, and mean shift algorithm is applied for finding the optimal object position. Based on the work of Comaniciu et al., Collins [12] introduced a method which tracks the object scale changes based on local maxima of differential scale-space filters. However, it requires high computational costs and cannot handle the rotation changes of the object. Zivkovic and Krose [28] regarded the mean shift as an EM-like algorithm, thereby presented the EM-shift algorithm which simultaneously estimates the position of the local mode and the covariance matrix that can approximately describe the shape of the local mode. Yilmaz [13] introduced a new kernel-based tracking using asymmetric kernels with adaptive scale and orientation selection method which is currently achieved by greedy algorithms.

Studies on online learning algorithms originated in computational learning community. The initial algorithms train several experts based on the labeled samples arriving sequentially and later combine the predictions of these experts to categorize any new example. These algorithms are popular to the machine learning researchers like the weighted majority algorithm [14] or winnow algorithm [15]. Both the weighted majority and winnow algorithms work as a committee of hypotheses to classify target samples. The popular (offline) AdaBoost classifier [16] resembles these classifiers in the sense that it also combines several “weak” hypotheses in classifying new observations. An online version of the boosting classifiers has been proposed in the study by Oza and Russell [17].

Online learning algorithms are widely used for object tracking. Collins et al. [24] developed an online feature selection mechanism using the two-class variance ratio to find the most discriminative RGB color combination in each frame. Avidan [3] proposed an ensemble tracking, in which an ensemble of weak classifiers is first trained online to distinguish between the object and the background for visual tracking. Parag et al. [2] advocated an online boosting algorithm where parameters of weak classifiers are updated using weighted linear regressors to minimize the weighted least square error (LSE). Wang et al. [19] presented a discriminative model that casts appearance modeling and visual matching into a single objective for visual tracking. Kuo et al. [18] introduced an approach for online learning of discriminative appearance models for robust multi-target tracking in a crowded scene from a single camera, which consists of two components: online sample collection and appearance model learning. Though these methods usually have the capability to select good features for tracking, they need correctly labeled samples to train and update classifiers. Gu et al. [21] utilized nearest neighbor (NN) classifi-
er to achieve stability and plasticity during tracking the targets of changing appearance. NN requires no training other than data collection, and it is efficient when the size of the data sample is small.

3. KERNEL RECURSIVE LEAST SQUARE

3.1 Kernel Recursive Least Square

The recursive least squares (RLS) algorithm [9] is an efficient online solution method for obtaining the least squares linear predictor by minimizing the weighted cost function. Although the least mean squares exhibit fast convergence, this benefit comes at the cost of high computational complexity. In this paper, we apply KRLS to train weak classifiers (base learners). Traditional discriminative learning algorithms for training a binary classifier require a training data set \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \), where \( x_i \in \mathbb{R}^d \) is the sample (a feature vector computed from a pixel), and \( y_i \in \{+1, -1\} \) is the label (+1 for foreground, −1 for background). In standard recursive least square algorithm, the loss function is defined as

\[
L(\beta) = \sum^n_{i=1} \omega_i (f(x_i) - y_i)^2 = (X\beta - Y)^TW(X\beta - Y),
\]

where \( f(x_i) = x_i^T\beta_0 + \beta_0 \) is a weak classifier, \( \beta = \begin{bmatrix} \beta_0^T & \beta_1^T \end{bmatrix}^T \), \( Y = (y_1, y_2, \ldots, y_n)^T \), and \( X \in \mathbb{R}^{n \times (d+1)} \) denotes \( n \) samples, and \( W \) is a diagonal matrix with sample weight \( \omega_i \) on its diagonal, respectively. We can obtain the optimal solution for \( \beta \) by minimizing the loss function (1). The number of equations \( n \) may be much larger than the dimension of \( \beta \) and may increase as we gather more observations, and hence storage of the actual data has a significant impact on the performance of the algorithm. We employ an efficient recursive updating method to handle this problem. Details will be demonstrated in Section 3.2.

We replace traditional linear classifiers with the kernel classifiers to classify the nonlinear separable data. The kernel classifier typically outputs a predictor given by \( f(x) = \sum^n_{i=1} \alpha_i k(x, x_i) \) where \( k(\cdot) \) is a Mercer kernel and \( \alpha_i \) is the corresponding coefficient. Different types of the kernel (e.g., Gaussian kernel and polynomial kernel) can be selected in our algorithm.

Supposing \( \phi(x) \) is the mapping of \( x \) in Hilbert space by the kernel; then, Eq. (1) turns to be

\[
L(\beta) = (\Phi\beta - Y)^TW(\Phi\beta - Y),
\]

where \( \Phi = (\phi(x_1), \phi(x_2), \ldots, \phi(x_n))^T \). Furthermore, we can verify \( \beta = \sum_{i=1}^n \alpha_i \phi(x_i) = \Phi^T\alpha \), where \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T \). Denoting \( K = \Phi\Phi^T \), we have

\[
L(\alpha) = (\Phi\Phi^T\alpha - Y)^TW(\Phi\Phi^T\alpha - Y) = (K\alpha - Y)^TW(K\alpha - Y),
\]

where \( [K]_{i,j} = k(x_i, x_j) \) is the full kernel matrix. We are able to obtain \( \alpha = (K^TWY)^{-1}K^TY \).
by minimizing Eq. (3).

3.2 Online Sparsification

The computational cost for traditional kernel method grows linearly as the samples increase. In order to reduce computation complexity, online sparsification is required. For a new sample \( x_t \), we have to decide whether to add it into the data dictionary or discard it [9]. Assuming that at time step \( t \), we have \( n_t - 1 \) training samples and a dictionary consisting of a subset of the training samples \( \{x_i\}_{i=1}^{n_t-1} \). If the new sample \( x_t \) satisfies

\[
\sum_{i=1}^{n_t-1} a\phi(x_i) - \phi(x_t) \leq \nu,
\]

(4)

We call Eq. (4) the approximate linear dependence (ALD) condition, where \( \nu \) is an accuracy parameter determining the level of sparsity. To avoid adding the training sample \( x_t \) into the dictionary, we need to find the optimal coefficients \( a = (a_1, a_2, \ldots, a_{n_t})^T \) satisfying the ALD condition. By simple derivation, we obtain \( a = \tilde{K}^{-1} \tilde{k}(x_t) \), where \( \tilde{K} = [k(\tilde{x}_i, \tilde{x}_j)]_{i,j=1}^{n_t} \), \( \tilde{k}(x_i) = k(\tilde{x}_i, x_t) \), \( i = 1, 2, \ldots, n_t \). If \( \delta_i > \nu \), we need to expand the current dictionary with \( x_t \cdot D_t = D_{t-1} \cup \{x_t\} \) and \( m_t = m_{t-1} + 1 \). By choosing \( \nu \) small enough, the approximation error can be small enough. Consequently, for every time step \( t \), we have

\[
\Phi_t = A_t \Phi_t^\pi + \Phi_t^{\pi \pi},
\]

(5)

where \( \Phi_t = (\phi(x_1), \phi(x_2), \ldots, \phi(x_t))^T \), \( \Phi_t^\pi = (\phi(\tilde{x}_1), \phi(\tilde{x}_2), \ldots, \phi(\tilde{x}_t))^T \), \( [A_t] = a_{ij} \) and \( \Phi_t^{\pi \pi} \)
denotes the residual component vector. By omitting $\Phi_{t}^{res}$, the corresponding approximation of full kernel matrix is given by

$$K_{t} \approx A_{s}^{T} \tilde{K} A_{s}.$$  \tag{6}

It means that the full kernel $K_{t}$ can be replaced with a "reduced" form. Defining $a_{r} = A_{s}^{T} \alpha$ and substituting Eq. (6) into Eq. (3), we have

$$L(a_{r}) = (A_{s}^{T} \tilde{K} A_{s}^{T} a_{r} - Y_{r})^{T} W(A_{s}^{T} \tilde{K} A_{s}^{T} a_{r} - Y_{r}).$$  \tag{7}

We are able to get $a_{r} = (A_{s}^{T} \tilde{K} W A_{s})^{-1} A_{s}^{T} W Y_{r}$ by minimizing Eq. (7). Denoting $P_{r} = A_{s}^{T} W A_{s} \in R^{m \times m}$ and $S_{r} = A_{s}^{T} W Y_{r} \in R^{m \times 1}$, $a_{r}$ can be rewritten as

$$a_{r} \approx (A_{s}^{T} W A_{s})^{-1} P_{r} S_{r}.$$  \tag{8}

The KRLS algorithm with online sparsification is summarized in Algorithm 1. The sparser the solution of a kernel algorithm is, the less time and memory it needs in the operational stage of the kernel methods. Furthermore, sparsification is related to generalization ability, which is a desirable property in machine learning algorithms [9].

4. KRLS BASED ONLINE BOOSTING FOR TRACKING

4.1 Tracking Approach

The target is manually marked in the first frame with a rectangle $R_{i}$ and a larger rectangle $R_{0}$ is selected around the inner rectangle $R_{i}$ to mark the pixels of the background. We treat all the pixels in rectangle $R_{i}$ as positive examples ($y_{i} = +1$) and all the pixels in $R_{0} - R_{i}$ are treated as negative examples ($y_{i} = -1$). The feature vector extracted for a sample is the R, G and B component of a pixel, which is simple but effective in our method. We train a strong classifier $H_{t}(x) = \sum_{k} c_{t} f^{k}(x)$ where $c_{t}$ is the weight of $f^{k}(x)$ by AdaBoost algorithm [16] using these samples. It is a linear combination of several weak classifiers $f^{k}(x) : R^{k} \rightarrow \{-1, +1\}, k = 1, 2, ..., T$ which are trained using KRLS with online sparsification introduced in Section 3. When given a new video frame, we classify each pixel in a larger region centered at previous target location using this strong classifier; and then, a confidence map is generated. We run mean shift algorithm on this confidence map, and the new position of target will be obtained easily.

However, AdaBoost is sensitive to outliers, and inaccurate samples will influence the learning of classifiers. Hence, an outlier rejection scheme [7] is needed. We use an approach which treats too "difficult" examples as outliers and change their labels [3]. The standard of the selection of samples can be rewritten as

$$y_{i} = \begin{cases} +1 & \text{inside}(r_{j}, p_{i}) \land (a_{r} < \eta) \\ -1 & \text{otherwise} \end{cases}$$  \tag{12}

where $r_{j}$ is the current rectangle, $p_{i}$ is the pixel position of example $i$ and $\text{inside}(r_{j}, p_{i})$ is
a predicate that is true if pixel \( p_i \) is inside rectangle \( r_j \), \( \omega_j \) is the weight of the pixel \( p_i \) after running the strong classifier and \( \eta \) is some predefined threshold, respectively. That is, pixels inside the rectangle are assumed to be positive examples, unless they are too “difficult” to classify and then their labels are changed to negative. After this outlier rejection scheme, the confidence maps are much more accurate leading to a more stable tracking result.

4.2 KRLS Based Online Boosting

During tracking, target appearance suffers different challenges, such as illuminations changes and occlusions. Therefore, the strong classifier ceases to be effective after several frames. In this paper, the classifier is updated by KRLS based online boosting. The underlying idea for development of online boosting classifier is to learn incrementally. As we mentioned earlier, the coefficients of weak classifiers should be updated with the new arriving samples. Motivated by the work of Parag [2], we use a new online learning algorithm in which the parameters of the weak classifiers are updated in accordance with the new data subset presented to the online boosting process at each time step.

Let \( X_t \) denote the samples obtained from frame \( t \), \( \alpha_t \) be the linear coefficients of the regressor learned from \( X_t \). For a new subset \( X_v \), the linear coefficients \( \alpha_{t+1} \) should be learned on \( X_{t+1} = [X_t^T, X_v^T] \) and \( Y_{t+1} = [Y_t^T, Y_v^T] \) by the KRLS algorithm as follows,

\[
\alpha_{t+1} = \tilde{K}_{t+1}^{-1} P_{t+1}^{-1} S_{t+1}. \tag{9}
\]

Here \( P_{t+1} = A_{t+1}^T W_t A_{t+1} \), \( S_{t+1} = A_{t+1}^T W_t Y_{t+1} \), \( W_{t+1} = \begin{pmatrix} W_t & 0 \\ 0 & W_v \end{pmatrix} \), where \( W_v \) is a diagonal matrix having the boosting weights of the new samples on its diagonal. We can easily verified that if we keep the data dictionary constant in this period, and the two parts required for computing \( \alpha_{t+1} \) in Eq. (7) can be decomposed and expressed as a recursive summation of previous and new samples,

\[
\begin{align*}
P_{t+1} &= A_{t+1}^T W_t A_{t+1} + A_{t+1}^T W_v A_{t+1} = P_t + P_v, \\
S_{t+1} &= A_{t+1}^T W_t Y_{t+1} + A_{t+1}^T W_v Y_{t+1} = S_t + S_v. \tag{10}
\end{align*}
\]

Thus this approach avoids the common issue of how many weak classifiers to be replaced artfully. From Eq. (8), we notice that all the new samples in \( X_v \) have equal importance with the old ones in \( X_t \), and hence have the same contribution towards the update of the weak classifier. As a result, after several steps, the weak learner will become biased to the recent samples and gradually lost the contributions of the first set of examples. If we update the classifier with the tracking result which is not accurate, the performance of tracker will be degraded, especially when the occlusion occurs. So we use a temporal weight \( \rho_t \) to tackle this problem. It is expressed as

\[
\begin{align*}
P_{t+1} &= \rho_t P_t + P_v, \\
S_{t+1} &= \rho_t S_t + S_v. \tag{11}
\end{align*}
\]
The speed of update of classifiers can be controlled through changing the value of \( \rho \). Generally, when \( \rho > 1 \), we expect to focus on the old samples and slow down the update speed. The KRLS based online boosting algorithm is shown in Algorithm 2.

**Algorithm 2**: KRLS based online boosting

**Input**: new dataset \( X_v \) and the label \( Y_v \).

**Initialization**: Set weights \( \{\omega_i\}_{i=1}^N \) to \( 1/N \).

for \( k = 1, 2, \ldots, n \),

1. Normalize the weights \( \{\omega_i\}_{i=1}^N \).
2. Compute \( P_v \) and \( S_v \) according to Algorithm 1. Learn \( \alpha \) with Eq. (9) and Eq. (11), and get classifier \( f^k(x) \).
3. Calculate Classifier error \( \varepsilon^k = \sum_{i=1}^N \omega_i | f^k(x_i) - y_i | \) and classifier weight \( c^k = 1/2 \log((1-\varepsilon^k)/\varepsilon^k) \).
4. Update the weights \( \omega_i^{k+1} = \omega_i^k \exp(c^k y_i f^k(x_i))/Z^k \).

end for

**Output**: The strong classifier \( H(x) = \sum_{k=1}^T c^k f^k(x) \).

### 5. EXPERIMENTAL RESULTS

In this section we show the experimental results of the proposed visual tracking algorithm in real-world scenarios. The videos were recorded in indoor and outdoor environments in different formats (color or grayscale). We use Overlap Ratio in the VA-CE-CLEAR protocol [10] as the objective performance evaluation. The performance of our method is compared with the linear recursive least square tracker (referred to as linear RLS tracker) [2], ensemble tracking [3], MIL tracking [5] and online multiple support instance tracking (OMSIT) [11]. The linear RLS tracker also uses boosting adaptive weak classifiers and R, G and B features; however, the weak classifiers are trained with linear regression without kernel.

The optimal total number of weak learners for our tracker in all the tracking examples is 20, which is sufficient for accuracy as well as efficiency needs. We update the dictionary every 10 frames. The strong classifier can be updated just using the samples in current frame instead of keeping all the data before. The online sparsification threshold \( \nu \) ranges from 0.7 to 1, and the selection depends on the number of samples for learning in different images. The proposed approach was implemented in Visual Studio 2010 on an Intel Core2 2.53 GHz processor with 2GB RAM. Average run time for our method is about 4100-4500ms per frame for all the experiments. No code optimization was performed.

#### 5.1 Qualitative Comparison

**Face & Tiger Sequence**: There are many challenges in these two sequences. The face sequence exhibits heavy occlusions, and the area of occlusion is bigger than a half of the face, as shown in Fig. 2. The tiger sequence in different poses contains frequent occlusions and fast motion (which causes motion blur), which is shown in Fig. 3. The ensem-
Fig. 2. Tracking results on face sequence. Red box: our method, yellow box: linear RLS tracker, blue box: ensemble tracker.

Fig. 3. Tracking results on tiger sequence. Red box: our method, yellow box: linear RLS tracker, blue box: ensemble tracker.

ble tracker moves with the book at the beginning of tracking in faces sequence. Although it comes back to the face in the end due to the update of tracker, it is not stable and precise. The linear RLS tracker drifts when the occlusion becomes heavier and heavier, especially in the tiger sequence. In contrast, our method performs very well through both of the sequences, and the tracker adapts to object appearance changes very rapidly and tracks it correctly.

David sequence: In this sequence, David’s face undergoes illumination changes when he moves. The linear RLS tracker and ensemble tracker lose the target after tracking it for a while. However, our tracker gives a precise position of the face even if the illumination changes incessantly and it clearly exhibits the robustness of our method for environmen-
tal changes. Some tracking result frames are given in Fig. 4. We notice that although the color feature we used is relatively simple, especially in grayscale images, where the values in three RGB channels is the same, the tracker performs excellently in different sequences, no matter color or grayscale, which suggests the superiority of our method.

![Tracking results on David sequence. Red box: our method, yellow box: linear RLS tracker, blue box: ensemble tracker.](image_url)

**Football & Car sequence**: These two sequences test the tracker on complex background. There is much noise and background clutter in the images, and the car and football player look very similar to the environment around it. In football sequence, the linear RLS tracker easily gets stuck to the background or causes a drift, and the ensemble tracker performs relatively better and it tracks the football player closely all the time except last few frames, as Fig. 5 shows. In car sequence, although the linear RLS tracker and ensemble tracker can follow the target most of frames in the sequence, they do not provide an exact and stable position for the target, which is shown in Fig. 6. Different with above two methods, our tracker locks on the target and gives an accurate position throughout the whole sequences. It proves that the weak classifier trained with kernel RLS is good at dealing with complex background as we motioned earlier, and this is the main reason why we apply kernel RLS in boosting framework.

### 5.2 Quantitative Comparison

Overlap Ratio defines the similarity between the track results and the ground truth. We manually mark the ground truth and compute the Overlap Ratio frame by frame in face sequence, David sequence and car sequence, and then draw curves with MATLAB for the convenience of analysis. From Fig. 7, we can see Overlap Ratio of our tracker is higher than that of the linear RLS tracker and MIL tracker most of the time, which suggests our method provides a much more accurate and constant tracking. Additionally, we prove that online sparsification can reduce data redundancy and increase the computational speed, which ensures real-time tracking.
Fig. 5. Tracking results on Football sequence. Red box: our method, yellow box: linear RLS tracker, blue box: ensemble tracker.

Fig. 6. Tracking results on Car sequence. Red box: our method, yellow box: linear RLS tracker, blue box: ensemble tracker.

Fig. 7. Comparison of Overlap Ratio on three video sequences; Red solid curve: our method; Blue dashed curve: linear RLS tracker. Green dotted curve: MIL tracker.
Table 1 displays number of frames exactly tracked by the proposed method and other methods on the aforementioned video sequences. We calculate the number of frames manually from the tracking output images. The standard of judging correctly tracking is that, in any frame, if more than 30% of the tracking window does not contain the ground truth, we label this result as tracking failure. Our method can track the target for almost the full length of the sequences, where the other trackers perform badly.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Object</th>
<th>Linear RLS</th>
<th>Ensemble tracking</th>
<th>OMSIT</th>
<th>MIL</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>David</td>
<td>Face</td>
<td>39/178</td>
<td>60/178</td>
<td>124/178</td>
<td>82/178</td>
<td>172/178</td>
</tr>
<tr>
<td>Football</td>
<td>person</td>
<td>9/45</td>
<td>41/45</td>
<td>15/45</td>
<td>45/45</td>
<td>42/45</td>
</tr>
<tr>
<td>Car</td>
<td>car</td>
<td>170/201</td>
<td>121/201</td>
<td>56/201</td>
<td>23/201</td>
<td>201/201</td>
</tr>
<tr>
<td>tiger</td>
<td>tiger</td>
<td>8/56</td>
<td>17/60</td>
<td>29/60</td>
<td>25/60</td>
<td>56/60</td>
</tr>
</tbody>
</table>

5.3 Discussion

The classification power of weak classifier using linear regression in low dimensional space is poor. Inspired by the good performance of kernel methods in [4, 7, 20] etc., we argue that better classification and expressive ability of weak predictor will be obtained when we perform linear regression in high dimensional space using kernel methods. Therefore, KRLS [9] is used to train weak classifiers naturally. However, the computational complexity of traditional kernel method grows linearly as the samples increase, which makes it time-consuming to train weak classifiers. A good way to cope with this problem is to choose the most valuable sample. Online sparsification is applied to do this work in our paper. Finally, the AdaBoost framework is used to combine those weak classifiers to a strong classifier. Samples are classified using this strong classifier. In order to handle target appearance changes during tracking, the strong classifier is updated using KRLS based online boosting. Therefore, our tracking algorithm could obtain significant performance in most cases.

However, our tracker couldn’t work well in all situations. For example, it fails when there is a large pose change as is shown in Fig. 8. Our tracker drifts in this sequence. The reason is that the skater’s pose changes all the time, e.g., crouching down, standing up and outstretching hands, etc. However, our window for labeling the targets changes in neither the size nor shape. As a result, there will be many outliers in the samples used for

![Fig. 8. Failed tracking case for our method.](image-url)
KERNEL REGRESSION BASED ONLINE BOOSTING TRACKING

Fig. 8. (Cont’d) Failed tracking case for our method.

updating the classifiers with AdaBoost. It will degrade the strong classifier and make the tracking results incorrect. This issue has been encountered in self-bootstrap classifiers trained for object tracking. Based on the MIL [6], the classifier is updated with positive and negative bags rather than individual labeled examples.

6. CONCLUSION

In this paper, we have proposed a KRLS based online boosting for robust visual tracking. We employ the KRLS to train classifiers, which performs linear regression in a high-dimensional feature space on nonlinear separable dataset and is effective for complex background. In our boosting framework, adaptive nonlinear weak classifiers are applied, and the coefficients of weak classifiers are updated by the new samples rather than replacing them. Moreover, online sparsification is performed to filter samples in feature space to reduce the computational cost. The state of target is obtained by mean-shift algorithm running on the confidence map, which is computed by the strong classifier. The experimental results demonstrate advantages of our method, especially when the background is complex in video sequences.

REFERENCES

7. C. Burges, “A tutorial on support vector machines for pattern recognition,” Data


Hongwei Hu is currently a Ph.D. candidate in the Media Computing and Intelligent Systems Lab at the School of Computer Science, Beijing Institute of Technology. He has strong research interests in object tracking and computer vision.

Bo Ma received the Ph.D. degree in Computer Science in 2003 from Harbin Institute of Technology, Harbin, China. From 2004 to 2006, he was with the Department of Computer Science, City University of Hong Kong, Hong Kong, China, working on research projects in computer vision and pattern recognition. In June 2006, he joined the Department of Computer Science, Beijing Institute of Technology, Beijing, China, where he is now an Associate Professor. His current fields of interest include statistical pattern recognition, image object tracking and information fusion.
Yuwei Wu is currently a Ph.D. candidate in the Media Computing and Intelligent Systems Lab at the School of Computer Science, Beijing Institute of Technology. He received the M.S. degree in June of 2009, then, continued to pursue a Ph.D. degree in the same department from the September of 2009, under the supervision of Prof. Yun-de Jia. He has strong research interests in active contour models, level set methods, object tracking and computer vision.

Weizhang Ma received the M.S. degree in Computer Science from Beijing Institute of Technology in 2012. He is currently a Computer Vision Engineer in Infinova Corporation. His research interests include visual tracking, motion detection and video summary.

Kai Xie received the M.S. degree from the North Jiaotong University in 1987 and the Ph.D. degree from the School of Computer Science and Technology at Harbin Institute of Technology in 2005. He is now a Professor of Beijing Institute of Graphic and Communication. His research interests include statistical signal and image processing, convex optimization, machine learning and information theory.