RAPID VISUAL TRACKING WITH MODIFIED ON-LINE BOOSTING AND TEMPLATE MATCHING

Guibo Luo, Yuesheng Zhu, Qing Zhang
Communication & Information Security Lab
Shenzhen Graduate School
Peking University, Shenzhen, China
luoguibo@sz.pku.edu.cn, zhuys@pkusz.edu.cn, fjzhangqing@gmail.com

ABSTRACT
On-line learning is increasingly popular in visual tracking, but the challenge that it faced is how to adapt the appearance changes and avoid the drifting or missing track. In this paper, a fast visual tracking algorithm is proposed to make the tracker more accurate and stable in the complex variations situations like occlusions, illuminations and shape deformations. In the proposed algorithm, a modified on-line boosting method is developed to make the tracker more adaptive to variable scene and a template matching model is used to constrain the training samples, so that the accumulating errors in self-update learning can be alleviated effectively. In addition, an optimization process is used to reduce the computational burden. Our experimental results have demonstrated that compared with other on-line tracking methods the target can be accurately tracked with lower drifting error in the complicated environments by using the proposed algorithm. Moreover, the new tracker runs at 60 frames per second, and is suitable for the real-time catching and tracking.

KEY WORDS
visual tracking; on-line boosting; template matching

1. Introduction
On-line learning method has shown its effectiveness and adaptability in the visual tracking [1]. However, in the traditional on-line learning algorithms, the self-update procedure may accumulate errors in case of inaccurate tracking results, and lead to mistracking. This is called as the stability-plasticity [2] or template update problem [3], and becomes a key open issue in the on-line learning field. Recently, many researchers have proposed lots of methods to solve this problem. Grabner et al. [4] formulated the tracking as a semi-supervised learning problem by combining a non-adaptive prior and an on-line classifier. In the Grabner’s method, the labeled data at the first frame is used as a prior and the data collected during tracking is labeled by the prior. Though the drifting can be alleviated, it cannot adapt to the appearance changes and occlusions because the prior is fixed. Babenko et al. [5] used a multiple instance learning (MIL) algorithm to reduce the label jitter phenomenon during updates in the tracking. The basic idea of MIL paradigm is that during the training process the samples are packaged in bags rather than individual samples, that is, instances close to the current object position are considered as a positive bag and other instances as negative bags, and the classifiers are trained to discriminate these bags instead of samples. Kalal et al. [6][7][8] exploited a parallel framework including tracking, learning and detection (TLD) for long-term tracking. TLD uses a Lukas-Kanade-tracker(LKT) to guide the learning process of the object detector. This selective-update strategy can achieve promising results in long-term tracking. However, the LKT is not competent to track the objects with non-rigid transformations and partial occlusions.

In this paper, an improved on-line learning method is proposed to make the tracker more accurate and stable in the complex variations situations like occlusions, illuminations and shape deformations. Firstly, the feature selection method for on-line boosting is modified which allows the classifiers to be more adaptive to the appearance changes. The importance weight of a sample is reinitialized at every selection process and the error of the weak classifier is computed only from the importance weight of current classified samples. Compared with the algorithm in literature [1], the modified method can adapt to fast object appearance change and then achieve more accurate tracking results. Moreover, an on-line template matching model is used and several restrictive learning constraints are introduced to choose the on-line boosting training samples prudently so that the accumulation errors can be largely limited. Furthermore, the model is cascaded with on-line boosting, this complementary method can significantly alleviate the drifting while still be adaptive in visual tracking. The proposed algorithm is implemented both on PC and embedded system. Also the calculation process of the confidence map is optimized by using “batch processing” method so that new algorithm can speed up the computation about 300% compared with the typical algorithm [1].

The rest of the paper is organized as follows: the on-line feature selection is reviewed in Section 2; the proposed tracking algorithm is described in Section 3; the performance comparison among our method and the other
on-line based methods is shown in Section 4; and the conclusion is given in Section 5.

2. On-line Boosting and the Feature Selection

Boosting is an approach to produce a high accurate prediction rule by integrating some weak and rough learning algorithms into a strong one. On-line boosting is a real-time boosting algorithm in which the training data doesn't need to be given in advance. It is proved [14] that if on-line and off-line boosting are given the same training samples, then the final classifier returned by on-line boosting is the same with the one obtained by off-line boosting when the number of training iterations \( N \to \infty \). On-line boosting algorithm consists of three main parts, i.e., weak classifier, selectors, and strong classifier, and will be described as follows:

1) Weak classifier

Boosting is to construct a strong classifier from a set of weak classifiers which only perform a little better than random guess. A weak classifier \( h^{\text{weak}} \) is corresponding to a feature, the hypothesis generated by the weak classifier is based on the respond of feature by applying a learning algorithm.

In general, the features used for on-line boosting [1] are haar-like features [13]. Three features are used here, i.e., (a) two-rectangle, (b) three-rectangle and (c) four-rectangle, as shown in Figure 1. The feature value is equal to the sum of the pixels within white rectangles subtracted from the sum of pixels within the black rectangles.

![Figure 1. Three Types of Haar-like Features](image)

2) Selectors

In on-line boosting, the importance weight \( \lambda \) of a sample can be estimated by propagating it through a fixed set of weak classifiers [1]. On-line boosting is performed on the selected weak classifiers. The feature selection is to select the best \( N \) selectors \( \{h_{n}^{\text{weak}}, \ldots, h_{N}^{\text{weak}}\} \) from a given set of \( M \) weak classifiers with hypothesis \( H^{\text{weak}} = \{h_{1}^{\text{weak}}, \ldots, h_{M}^{\text{weak}}\} \), where a selected selector is corresponding to a weak classifier. Once a new training sample \( < x, y > \) arrives, the \( M \) weak classifiers are trained and the weak classifier with the smallest error will be selected by the criterion, \( h^{\text{weak}}(x) = \arg \min_{e} e(h_{n}(x)) \). The error of the weak classifier is: \( e = \lambda^{n} / (\lambda^{c} + \lambda^{n}) \), where \( \lambda^{c} \) and \( \lambda^{n} \) is the importance weight of correctly and wrongly classified samples, respectively, and these weights are affected by the samples seen so far. Finally, the importance weight \( \lambda \) and the corresponding voting weight \( \alpha_{n} \) of the selector are updated and then passed to the next selector, \( h_{n+1}^{\text{weak}} \). This procedure is iterated until \( N \) selectors are selected.

3) Strong classifier

A strong classifier is considered as a linear combination of the weak classifiers, and its corresponding hypothesis, \( H \), is a weighted vote of the \( N \) weak hypotheses, where \( \alpha_{n} \) is the weight assigned to \( h_{n}^{\text{weak}} \):

\[
H(x) = \text{sign}(\text{conf}(x))
\]

\[
\text{conf}(x) = \sum_{n=1}^{N} \alpha_{n} h_{n}^{\text{weak}}(x)
\]

where \( \text{conf}(\cdot) \) is a confidence measure, it ranges from -1 to 1, the higher value means the patch is more similar to the target object.

3. Proposed Tracking Algorithm

3.1 Modified On-line Feature Selection

The main idea of discriminative tracking method is to formulate the task as a classification problem and to continuously update the current classifier which optimally discriminates the target from the current background [11]. Therefore, in the on-line boosting algorithm, how to select a set of the good weak classifiers is a key to develop an effective tracker.

In the typical on-line boosting, the feature selection is to select the smallest error of the weak classifier, which is estimated from the weights of correctly importance \( \lambda^{c} \) and wrongly importance \( \lambda^{n} \) classified samples seen so far. Assumed the target at time \( t+1 \) only changes little compared to the target at time \( t \). In our tracking process, the classifiers for discriminating target object and background at time \( t \) will perform as nearly best to discriminate the target object and background at time \( t+1 \), and the classifiers only choose the samples from current frame and ignore the historical samples. These classifiers would make the tracking more accurate and more adaptive to fast object appearance change.

In our work, the importance weights are initialized at every selection process and the error of the weak classifier is estimated only from the importance weights of current classified samples. The importance weights are initialized as \( \lambda_{i} = 1,i = 1,2, \ldots, K \) in every frame, where \( k \) is the number of samples in current frame. The error of the weak classifier is computed by:

\[
e = \sum_{x} \text{abs}(h_{i}^{\text{weak}}(x_{i}) - y_{i})\lambda_{i}
\]

The principle of the algorithm is depicted as Algorithm 1.

In the typical on-line boosting the weak classifiers are updated and the best classifier is selected once a new training sample arrives, hence the computation complexity is high. In our proposed method the best classifier is selected from the five samples (a positive and four negative samples) of current frame and the weak
classifiers are updated so that the computation complexity can save 80% compared with the typical on-line boosting.

**Algorithm1:** Modified On-line Boosting for Feature Selection

Require: training samples \( <x_1, y_1>,...,<x_K, y_K> \)
where \( y_i = -1, 1 \) for negative and positive samples respectively.

Require: weak classifiers \( h_{i, weak},...,h_{M, weak} \) (initialized randomly)
Initialize the importance weight \( \lambda_i = 1, i = 1, 2,..., K \)

for \( n = 1, 2,..., N \) do
  \[ \lambda_i = \sum_{n=1}^{x} \lambda_i \]
  for \( m = 1, 2,..., M \) do
    \( e_{n,m} = 0 \)
    for \( i = 1, 2,..., K \)
      if \( h_{i, weak}(x_i) \neq y_i \) then
        \[ e_{n,m} = e_{n,m} + \lambda_i / \lambda_i \]
      end if
    end for
  end for
  \[ m^* = \arg \min_{m}(e_{n,m}) \]
  \[ e_n = e_{n,m^*}, h_{n,m^*} = h_{i, weak} \]
  if \( e_n = 0 \) or \( e_n > 1/2 \) then
    exit
  end if
  \[ \alpha_n = \frac{1}{2} \ln \left( \frac{1 - e_n}{e_n} \right) \]
  for \( i = 1, 2,..., K \)
    if \( h_{i, weak}(x_i) = y_i \) then
      \[ \lambda_i = \lambda_i \times \frac{1}{2(1-e_n)} \]
    else
      \[ \lambda_i = \lambda_i \times \frac{1}{2e_n} \]
    end if
  end for
end for

3.2 On-line Template Matching Model

To alleviate the drift problem, the object model [6] is applied in our on-line template matching model, and a set of normalized intensity patches \( M \) is used to represent the target and background, and defined as:
\[ M = \{ p_1^+, p_2^+, ..., p_m^+, p_1^-, p_2^-, ..., p_n^- \} \]

Where \( p_i^+ (i = 1, ..., m) \) are positive samples and \( p_j^- (j = 1, ..., n) \) are negative samples. In order to reduce redundancy in the model, only the patches which are much different with the existing samples in model can join the template matching model. In our method, the positive samples are: the samples in target object chosen in the first frame; or the samples in the patches of the tracking result in the tracking process, while the negative samples are: the samples far away from the target object in the first frame; or the samples whose confidence of on-line boosting is high but the positions are far away from that of the target in the tracking process.

In order to make the tracker of on-line boosting have the effective features of both plastic and stable, in this paper two modification are proposed (i) modify the feature selection process so that the tracker can adapt the fast appearance change of the object; (ii) use the on-line constrained template matching model in the update procedure to alleviate the drift problem. Also these two modules are combined in a cascade way as illustrated in Figure 2. In the proposed tracker, the first stage is the on-line boosting process, in which the object in search space is detected and the sample’s confidence is calculated, and then the patches whose confidence is smaller than a predefined threshold \( T \) (for example \( T = 0.5 \)), will be rejected. \( T \) can restrict the number of patches to enter the second stage. The second stage is the template match process. In this stage, the patches which output from the first stage and the distance to the target model is smallest will be picked up. The distance between two patches is defined by the normalized cross-correlation [7] as:
\[ distance(p_i, p_j) = 1 - NCC(p_i, p_j) \] (4)

And the distance of sample \( x_i \) to the target model \( M \) is defined as:
\[ distance(x_i, M) = \min_{x \in M} distance(x_i, x) \] (5)

Only when there are the patches which can pass through the two stages, the tracker can track the target successfully, otherwise lose the target.

![Figure 2. The block diagram of proposed tracking algorithm](image)

3.3 Batch Processing

In the tracking task, the confidence map of search space is calculated and analyzed [12] by the on-line boosting. The confidence of a patch is calculated as follows:
1) Locate the patch’s position,
2) Locate the feature’s position of the given patch,
3) Use integral image to calculate the rectangle features,
4. Experimental Results

The proposed tracker is implemented based on the on-line boosting algorithm by modifying the feature selection method and plugged template matching algorithm. Also the batch processing is used to speed up the computation. For several challenging video sequences, the performance comparison are made among our tracking method and OAB\(^1\), TLD\(^2\), MIL\(^1\) and SemiBoost\(^1\) approaches. Our experiments run on a computer with a 2.27GHz Intel Core i3 CPU and 4GB RAM for eight different challenging videos. All video frames are resized to 320x240 pixels. In our experiments, all parameters were set constant. The strong classifier consists of 50 selectors and the feature pool consists of 250 weak classifiers.

Table 1 and Figure 3 show the comparison of the average center location errors and the average performance of our tracker is the best. Figure 4 shows tracking results for some of the video scenarios. Row 1 depict occlusions and illumination change scenario: after the target is occluded (frame 175), OAB drifts away while SemiBoost wrongly switches to another similar object. When heavy illumination change happened (frame 358), TLD and SemiBoost get lost, MIL and OAB drifts away, only our tracker still locate the target accurately. The next sequence shows heavy occlusion case: OAB drifts away after frame 85 and cannot recover again. Although SemiBoost performs stable, it has 15% lost rate during this sequence. MIL and TLD drift little by little after frame 321. Our algorithm tracks the target accurately. The final sequence focus on scale change and 3D-motion: due to its scalable model and detector, TLD can recover after totally get lost, but it does not adapt to heavy appearance change (frame 173,278,336).

Table 2 shows the comparison results of computation speed and our optimized algorithm runs at 60fps, much rapidly than others do. Also our tracking algorithm is implemented on TI DM3730, a 1GHz ARM processor, and the tracking speed at 16fps is achieved, which demonstrates that our proposed method is suitable for the real-time catching and tracking on mobile terminals.

![Table 1. Comparison of Average Center Location Errors](image)

<table>
<thead>
<tr>
<th>Sequence</th>
<th>MIL</th>
<th>OAB</th>
<th>SemiBoost</th>
<th>TLD</th>
<th>ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car Chase</td>
<td>43</td>
<td>47</td>
<td>57</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>David</td>
<td>33</td>
<td>42</td>
<td>70</td>
<td>16</td>
<td>24</td>
</tr>
<tr>
<td>Faceocc</td>
<td>36</td>
<td>56</td>
<td>9</td>
<td>21</td>
<td>11</td>
</tr>
<tr>
<td>Faceocc2</td>
<td>22</td>
<td>31</td>
<td>77</td>
<td>19</td>
<td>11</td>
</tr>
<tr>
<td>Lemming</td>
<td>8</td>
<td>8</td>
<td>34</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>Panda</td>
<td>12</td>
<td>11</td>
<td>53</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Sylvester</td>
<td>15</td>
<td>15</td>
<td>18</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Tiger1</td>
<td>34</td>
<td>64</td>
<td>36</td>
<td>13</td>
<td>15</td>
</tr>
</tbody>
</table>

Note: Red: best result, Green: second best result.

![Table 2. Comparison of Computation Speed](image)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MIL</th>
<th>OAB</th>
<th>SemiBoost</th>
<th>TLD</th>
<th>ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed(fps)</td>
<td>4</td>
<td>13</td>
<td>8</td>
<td>8</td>
<td>60</td>
</tr>
</tbody>
</table>

From the results shown above, we can see that the proposed tracker is more adaptive and stable and the tracking runs more rapidly compared with other tracking algorithms.

5. Conclusion

In this paper, a rapid visual tracking algorithm is proposed, the on-line boosting method is modified to make the tracker adapt to the complicated environments, and a template matching model is employed to alleviate the drifting. Our analysis and experimental results have demonstrated the complementary of on-line boosting method and template matching model can make the tracker more stable and accurate. Furthermore, our algorithm has a superiority of speed over other on-line tracking algorithms, it runs at 60 frames per second and is suitable for the real-time catching and tracking.

References

Conf. on Computer Vision and Pattern Recognition, 2009, 983-990.

Figure 3. Center Location Errors for Eight Video Sequences

Figure 4. Tracking results of our tracker (red) and other on-line trackers. The sequences represented situations of (A) Car Chase: occlusions and illumination change, (B) Faceocc: heavy occlusions, (C) Panda: scale change and 3D-motion.