21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

Online Random Ferns for Robust Visual Tracking

Cong Rao, Cong Yao, Xiang Bai, Weichao Qiu and Wenyu Liu
Department of Electronics and Information Engineering
Huazhong University of Science and Technology, P. R. China
{*cong.rao, yaocong2010, xiang.bai, qiuwch*}*@gmail.com, liuwy@hust.edu.cn*

Abstract

Recently many appearance based visual tracking algorithms have been investigated, aimed at building robust appearance models against challenges brought by the varying appearance of the target as well as the unconstrained environment. More often adaptive appearance models were used to capture these variances over time, but this may sometimes result in losing the target (drifting) due to inappropriate update of the model. In this paper an online form of Random Ferns classifier is proposed to accomplish the task of robust appearance modeling with a constrained updating strategy against the potential incorrect update induced by runtime noise. Experiments on challenging benchmark video sequences have been conducted and improvement is observed when compared with recent state-of-the-art algorithms.

1. Introduction

Visual object tracking is a well studied yet significant problem in computer vision, since many practical applications demand a robust tracker so as to make a reliable analysis of the real world. The problem is also very challenging due to the variations happening to the target appearance as well as the noise existing in the unconstrained environment. To deal with the above uncertainties, conventional tracking algorithms typically make efforts in finding solutions out of 1) the appearance model, 2) the motion model, and 3) the searching strategy, as surveyed in [13]. And recently, there has been a great interest in the first aspect that aimed at building robust appearance models to better discriminate the target from the surrounding background. Adam et al. [1] represents the target as multiple fragments to better handle occlusions. Ross et al. [11] copes with appearance changes by performing incremental update on their PCA-based appearance model. As adaptive appearance models often induce error when inappropriately updated, Grabner et al. [6] alleviates the problem by making a constraint when updating. Others seek to find better learning methodology for appearance modeling via P-N constraint [8] or Multiple Instance Learning [2].

Inspired by the Random Ferns classifier for its nature of incremental training as well as its capability against overfitting [10], this paper extend it to the online form to accomplish the task of adaptive appearance modeling for robust visual tracking. Meanwhile, to limit the drifting problem caused by inappropriate update of the appearance model, a constrained model updating strategy is proposed with a notion of memory factor which controls how fast the appearance model could change during one single update. For the evaluated bench mark video sequences, our tracker has demonstrated robust and stable performance and improvement is observed when compared with recent state-of-the-art algorithms.

2. Online Random Ferns for Tracking

In this section we will introduce our Random Ferns [10] based visual tracking algorithm in details, in terms of strategies in appearance modeling, target searching, constrained model updating and image representation.

2.1. Appearance Model for Target Localization

In this subsection, we begin with a review over Random Ferns classifier, and then we will take an overview of the visual tracking system, introducing how we apply it for target appearance modeling and localization.

In the paradigm of visual object tracking, the target of interest is visually observed as an image patch at location $\boldsymbol{l}^*$ with size $w \times h$. Suppose $\boldsymbol{x} = \{x_1, x_2, \ldots, x_N\}$ denotes the N-dimensional feature vector extracted from a observed patch located at $\boldsymbol{l}$ and let c be some class label associated with it (although

978-4-9906441-1-6

in our case $c \in \{c_{target}, c_{backgnd}\}$, it can be naturally extended to the multi-class scenario). Then in order to locate the target, we need to find a location $\boldsymbol{l}^*$ of an image patch that maximizes certain kind of likelihood function which tells us how likely the patch is going to contain the target when we observe its feature. The Naïve Bayesian model [3] formulates this likelihood as

$$p(c|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|c)p(c)}{p(\boldsymbol{x})} = \frac{\prod_{i=1}^{N} p(x_i|c)p(c)}{p(\boldsymbol{x})}$$

where all features in $\boldsymbol{x}$ are assumed to be conditionally independent with each other.

The above naive assumption leads to a great reduction in the number of parameters needed to be estimated, but results in modeling the joint probability without considering the correlation between features. Thus, Random Ferns [10] is then introduced to make a compromise between the above oversimplification and full correlation modeling. It weakens the naive assumption by partitioning the features into M groups of size $S = N/M$, and then models the dependencies between intra-group features while preserving the assumption that inter-group features are conditionally independent. Under this semi-naive bayesian framework, the likelihood is consequently formulated as

$$p(c|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|c)p(c)}{p(\boldsymbol{x})} = \frac{\prod_{i=1}^{M} p(\boldsymbol{f}_i|c)p(c)}{p(\boldsymbol{x})} \quad (1)$$

where $\boldsymbol{f}_i = \{x_{(i,1)}, x_{(i,2)}, \ldots, x_{(i,S)}\}$ represents the i^{th} group (or *Fern*) of features. Features in a fern can be viewed as a subspace of the original feature space and they are selected in a manner of random permutation as described in the literature [10]. This kind of random subspace method has also been applied in the process of constructing Decision Forests [7] or Random Forests [4], which turn out to be effective in dealing with noise embedded in the features as well as robust against overfitting.

We utilize Random Ferns to model the appearance of the target of interest. That is, to feed the training data to the classifier and then estimate (or update) the parameters for each $p(\boldsymbol{f}_i|c)$ via Maximum Likelihood algorithm. Since we followed the original Random Ferns to use discrete-valued features (specifically, $x_i \in \{0, 1\}$), each class conditional probability $p(\boldsymbol{f}_i|c)$ is modeled as a Multinomial distribution [3].

Having constructed the appearance model at time t, we are then able to find the location $\boldsymbol{l}^*$ of target of interest in the incoming frame $(t+1)$ by maximizing the following objective function

$$\begin{aligned}\boldsymbol{l}^* &= \arg\max_{l} \frac{p(c = c_{target}|\boldsymbol{x})}{p(c = c_{backgnd}|\boldsymbol{x})} \\ &= \arg\max_{l} \frac{p(\boldsymbol{x}|c = c_{target})}{p(\boldsymbol{x}|c = c_{backgnd})} \cdot \frac{p(c = c_{target})}{p(c = c_{backgnd})} \\ &= \arg\max_{l} \frac{p(\boldsymbol{x}|c = c_{target})}{p(\boldsymbol{x}|c = c_{backgnd})}\end{aligned} \quad (2)$$

Since the prior of each category is fixed at time t, the ratio of category prior can be taken as a constant so that this term can be ignored in the above object function. Combining equation (1) and (2), the objective function for target localization is eventually derived as

$$\boldsymbol{l}^* = \arg\max_{l} \frac{\prod_{i=1}^{M} p(\boldsymbol{f}_i|c = c_{target})}{\prod_{i=1}^{M} p(\boldsymbol{f}_i|c = c_{backgnd})} \quad (3)$$

where $p(\boldsymbol{f}_i|c)$ can be calculated naturally with the appearance model in hand.

2.2. Training and Searching Strategy

Constructing the Training Set. Suppose the target has been located at $\boldsymbol{l}^{*(t)}$ at time t, then we take the following strategy to generate training samples for modeling the appearance of the target. The patch at $\boldsymbol{l}^{*(t)}$ is selected as the model of positive example, and we synthesize n_p different views of it as positive examples by applying random affine warping, random gaussian noise and blur to the selected patch, similar to the strategy taken in [9], [10]. This scheme can improve the generalization properties of classifier when encountered with small changes in view angle, scale or other runtime noise; the negative examples are obtained by randomly sampling the area away from $\boldsymbol{l}^{*(t)}$. That is, to randomly sample n_n image patches at locations in $\{\boldsymbol{l} : r_{smin} \leq \|\boldsymbol{l} - \boldsymbol{l}^{*(t)}\|_2 \leq r_{smax}\}$. The training set is then constructed by extracting features from the total $(n_p + n_n)$ image patches and associating them with respective class labels, and finally it is offered to update parameters of the appearance model in (1).

Searching for the Target. In the incoming frame $t+1$, we define the candidate examples that may contain the target as those image patches located within a specified radius r_{search} of center $\boldsymbol{l}^{*(t)}$, namely, the patches at *all* locations in $\{\boldsymbol{l} : \|\boldsymbol{l} - \boldsymbol{l}^{*(t)}\|_2 \leq r_{search}\}$. Having collected these candidates, we can evaluate (3) on all of them to predict $\boldsymbol{l}^{*(t+1)}$. This searching strategy indicates the form of our motion model which confines the range where the target could go in the next frame, and it also implies that our tracker is focusing on predicting the **location** of the target instead of considering the scale changes simultaneously.

2.3. Constrained Model Updating

Our appearance model of the target is designed to be updated incrementally aimed at coping with appearance changes while preserving robustness against short-term noise. We use a parameter m_f ($0 \leq m_f \leq 1$) named *memory factor* to control how fast the appearance model could change over time, which is similar to the notion of *forgetting factor* in previous literatures.

Let $\theta_{i,k,c_j} = p(\boldsymbol{f}_i = k|c = c_j)$ denote one of the parameters to be estimated for each multinomial distribution $p(\boldsymbol{f}_i|c)$ in (1), where we consider $\boldsymbol{f}_i$ to be equal to k if the base two number formed by the binary features in $\boldsymbol{f}_i$ is equal to k. With this convention, each $\boldsymbol{f}_i$ ($i \in \{1, 2, \ldots, M\}$) can take $K = 2^S$ different values, and θ_{i,k,c_j} indicates the class conditional probability of $\boldsymbol{f}_i$ taking the value k given its class label c_j ($j \in \{target, backgnd\}$). Extending the off-line method [10] for estimating each θ_{i,k,c_j}, we propose to update the parameter incrementally by taking the following strategy

$$\theta^{(t)}_{i,k,c_j} = \frac{\sum_{t'=0}^{t-1} {m_f}^{t-t'} n^{(t')}_{i,k,c_j} + n^{(t)}_{i,k,c_j}}{\sum_{t'=0}^{t-1} {m_f}^{t-t'} n^{(t')}_{i,c_j} + n^{(t)}_{i,c_j}} \qquad (4)$$

where $n^{(t)}_{i,k,c_j}$ is the number of training samples of class c_j that support $\boldsymbol{f}_i = k$, and $n^{(t)}_{i,c_j}$ is the number of training samples of class c_j used to update the parameters related to $\boldsymbol{f}_i$. The superscript t of all notations indicates they are varying with time t, except for that of m_f which means the power operation. To better illustrate this updating strategy, we then proceed to denote the numerator and denominator of (4) as $N^{(t)}_{i,k,c_j}$ and $N^{(t)}_{i,c_j}$ respectively, thus (4) can be rewritten as

$$\theta^{(t)}_{i,k,c_j} = \frac{N^{(t)}_{i,k,c_j}}{N^{(t)}_{i,c_j}} = \frac{m_f N^{(t-1)}_{i,k,c_j} + n^{(t)}_{i,k,c_j}}{m_f N^{(t-1)}_{i,c_j} + n^{(t)}_{i,c_j}} \qquad (5)$$

The above formulation gives the intuition that m_f is meant to control how much $\theta^{(t)}_{i,k,c_j}$ is going to be affected by the previous collected training examples. It can be verified that $\sum_k \theta^{(t)}_{i,k,c_j} = \mathbf{1}$ since $\sum_k n^{(t)}_{i,k,c_j} = n^{(t)}_{i,c_j}$, and it can also be observed that $\theta^{(t)}_{i,k,c_j}$ can be calculated iteratively, which makes the tracker computationally efficient since it's not necessary to keep a record of training examples accumulated at all times. A fixed $m_f = 0.85$ was used in all our experiments.

2.4. Image Features

Most learning based algorithms often require vector-like input as a portrait for the concrete object in the real world, and a high quality representation of object often plays an important role in the learning process. Haar-like feature has shown good properties in the application of object detection [12]. Babenko et al. [2] extended it by randomly generating the parameters of the feature and achieved good performance in the visual tracking paradigm. Therefore, we utilize the same kind of feature proposed in [2], but make the following modification to generate binary features required by Random Ferns [10]: we randomly generate N such Haar-like features and use a threshold of zero for the binarization of each feature. The technique of integral image [12] is also applied for efficiently computing the features.

Table 1. Average Center Location Error

Sequence	OAB	SBT	FT	MIL	Ours
Snack Bar	15.8	57.2	34.7	**13.2**	**10.7**
Cou. Book	**8.2**	65.1	55.9	**15.3**	20.7
David Ind.	41.6	44.4	70.5	**28.7**	**18.6**
Surfer	21.6	9.3	140.0	**8.6**	**6.0**

Note: **bold blue** font indicates best performance and **bold cyan** font indicates second best.

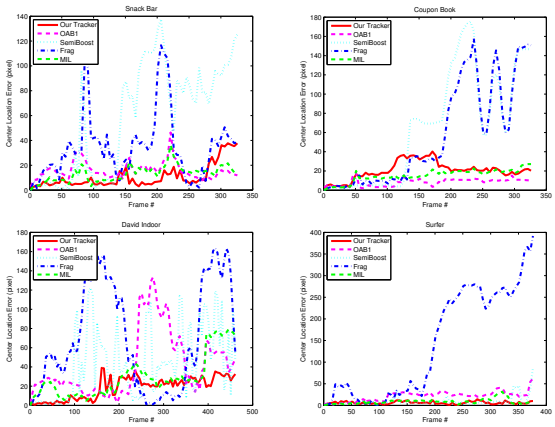

Figure 1. Center Location Error per Frame

3. Implementation and Experiments

The experiments were conducted on several benchmark video sequences[1], which are challenging as they contain variations like rotation, scale changes, motion blur, illumination changes or partial occlusion. To evaluate the relative performance of the proposed tracker, we also compare our results with four other algorithms,

[1] http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml

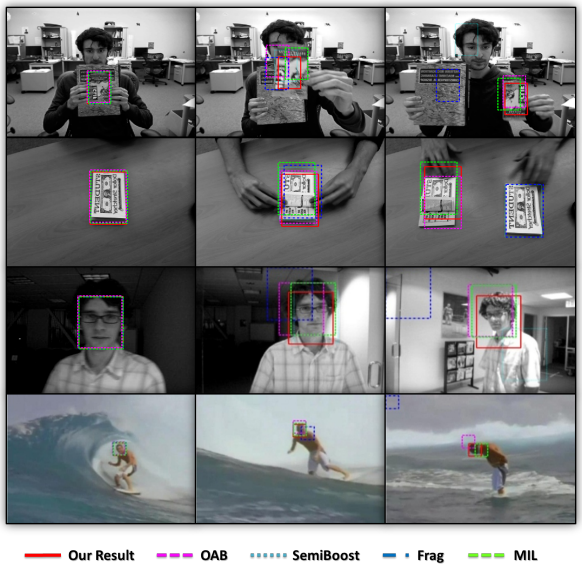

Figure 2. Screen Shot of Tracking Results

namely Fragment-based Tracker [1], OAB [5], SemiBoost Tracker [6] and MILBoost [2], using the code provided by respective authors.

Parameters Configurations. Parameters were set fixed for all the experiments to demonstrate the robustness of our tracker. In each frame, we generate $n_p = 128$ different views of the positive example model and randomly choose $n_n = 128$ negative examples within radius between $r_{smin} = 6$ and $r_{smax} = 50$ of the previously predicted target location. The search radius r_{search} is set as 40. We use $M = 128$ ferns with a fern size S of 8 for the Random Ferns classifier, resulting in a total number of features N of 1024.

Performance Evaluation. Conventional qualitative evaluation of a tracker is performed by calculating center location error of each frame and averaging them to summarize the overall location error on certain video sequence. We also follow this scheme to make performance evaluation. Due to the randomization strategy taken in our algorithm, we run the proposed algorithm for five times on each video sequence and take the average performance to see if it is uniformly outperforming. The quantitative results are summarized in Table 1 and plots of the center location error for each frame are depicted in Figure 1. It is shown that for the evaluated sequences, our tracker has demonstrated competitive (often superior) performance compared with those state-of-the-art algorithms. An intuitive illustration on the performance of the different tracking algorithms is given in Figure 2. The implemented tracker has achieved a near real-time speed (about 10 fps) on a normal desktop machine and further speed-up can be expected if parallel programming techniques (like multithreading or GPU) are engaged.

4. Conclusion

In this paper, we proposed a novel appearance based tracking algorithm with a constrained model updating strategy aimed at limiting the drifting problem induced by inappropriate update of the appearance model. An online form of Random Ferns classifier is also derived to adaptively model the appearance of the target with randomized binary Haar-like feature. Experiments have shown that improvement is made when compared with recent state-of-the-art algorithms.

Acknowledgements. The authors would like to thank those dedicated reviewers for their valuable comments and suggestions. This work was supported by the National Natural Science Foundation of China #60903096 and #61173120.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, 2006.

[2] B. Babenko, M. H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *TPAMI*, 33(8):1619–1632, 2011.

[3] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, 2006.

[6] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008.

[7] T. K. Ho. The random subspace method for constructing decision forests. *TPAMI*, 20(8):832–844, 1998.

[8] Z. Kalal, J. Matas, and K. Mikolajczyk. P-n learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, 2010.

[9] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *CVPR*, 2005.

[10] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *TPAMI*, 32(3):448–461, 2010.

[11] D. Ross, J. Lim, R. S. Lin, and M. H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1):125–141, 2008.

[12] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[13] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006.