2012 IEEE International Conference on Multimedia and Expo

THE EXTENDED CO-LEARNING FRAMEWORK FOR ROBUST OBJECT TRACKING

Chen Gong, Yang Liu, Tianyu Li, Jie Yang
Department of Automation, Shanghai Jiao Tong University, and Key Laboratory of System Control and Information Processing, Ministry of Education of China
Shanghai, China
goodgongchen@sjtu.edu.cn

Xiangjian He
Faculty of Engineering & Information Technology
University of Technology, Sydney PO Box 123, Broadway 2007, Australia
Sydney, Australia
Xiangjian.He@uts.edu.au

Abstract—**Recently, object tracking has been widely studied as a binary classification problem. Semi-supervised learning is particularly suitable for improving classification accuracy when large quantities of unlabeled samples are generated (just like tracking procedure). The purpose of this paper is to fulfill robust and stable tracking by using collaborative learning, which belongs to the scope of semi-supervised learning, among three classifiers. Different from [1], random fern classifier is incorporated to deal with 2bitBP feature newly added and certain constraints are specially implemented in our framework. Besides, the way for selecting positive samples is also altered by us in order to achieve more stable tracking. Algorithm proposed in this paper is validated by tracking pedestrian and cup under occlusion. Experiments and comparison show that our algorithm can avoid drifting problem to some degree and make tracking result more robust and adaptive.**

Keywords-tracking; semi-supervised learning; collaborative learning; 2bitBP feature; random fern classifier

1. INTRODUCTION

Object tracking is widely used in a variety of situations such as intelligent surveillance, scene understanding and behavior analysis. The task of tracking is to locate moving object in frames of a video sequence. So far, the existing tracking algorithms can be divided into three main types, namely:

(I) Motion based: By capturing and analyzing motion information, moving object can be easily segmented. Existing typical means are frame-difference method, optical flow[2], etc. These algorithms are easy to implemented, while slight luminance change or partially occlusion will lead to damage.

(II) Searching and matching based: Searching the predefined region and trying to find which sub-region matches the moving object best by virtue of some existing similarity measures. [3][4][5] belong to this class. However, the process of searching and matching is time-consuming. For the purpose of solving this problem, some advanced searching methods such as mean-shift[6] have been proposed. An obvious shortcoming of mean-shift is that only single feature information(color distribution) is used, so stable and robust tracking is hard to achieve.

(III) Estimation and prediction based: given the location of moving object in current time t, the goal is to estimate its location in time $t+1$. Some typical algorithms that belong to this type are Kalman filter and particle filter[7]. But they are sensitive to environmental change and need relatively heavy calculation burden.

In one word, all the methods mentioned above are short of adaptive ability because of lacking update or learning procedure. For addressing this problem, a new idea has been proposed in recent years that the tracking can be regarded as a classification problem, i.e., the process of tracking is to distinguish object to be tracked from background during which the classifier is updated continually. Avidan[8] opened the door for this idea by using support vector machine. Kalal et al. proposed tracking-learning-detection(TLD) method[9] for face tracking. [10] adopted an on-line boosting classifier that selects features to discriminate the moving object from the background, but every time the classifier is updated, an error might be introduced, which will probably cause drifting problem.

In order to improve tracking robustness and adaptivity, semi-supervised learning was applied to make full use of both labeled and unlabeled data collected in every frame. There are many semi-supervised learning algorithms such as self-learning, co-learning, graph-based, semi-supervised support vector machine(S3VM), etc. [11][12] perform tracking via graph-based semi-supervised learning successfully. [13] has proposed a co-training framework to combine one global generative tracker and one local discriminate tracker. In [1], the authors come up with a co-tracker who trains two SVM classifiers with RGB color histogram, histograms of oriented gradients(HoG) [14] features and their classification results are combined by weight sum rule. These two classifiers are updated in a collaborative fashion simultaneously. However, through our experiments, we have found that tracking error often arises if environmental condition modifies abruptly. We consider that this problem may be caused by "disagreement" between two classifiers. When this situation happens, the weight sum rule will probably hurt tracking result, so the idea of employing the third classifier to address this "disagreement" is straightforward.

In this paper, we have adopted three classifiers to form a robust tracking framework. The main differences or improvements between our work and [1] are:

(I) One more classifier called Random Fern[15] and corresponding Local 2bit Binary Patterns(2BitBP)[16] feature are elaborately added.

(II) In each frame, the number of generated positive sample is not fixed to one, which differs from [1] in which only one sample corresponding to the tracking object is assigned positive label.

(III) In order to prevent drifting problem, certain constraints are designed to guarantee the accuracy of classification.

The rest of this paper is organized as follows. Section 2 briefly reviews some basic knowledge which will be used in later sections. Section 3 describes our algorithm in detail and finally presents the flowchart of our tracking framework. Tracking results on some typical sequences and comparison with several prevalent tracking methods are shown in section 4. At last, conclusion can be drawn in section 5.

2. PRELIMINARIES

2.1. Semi-supervised Learning and Co-learning

Semi-supervised learning is somehow between supervised learning and unsupervised learning. This learning strategy is particularly suitable for situations where a large number of unlabeled samples exist. If certain assumptions are satisfied, these unlabeled samples are greatly beneficial to improve classification result even though their labels are unknown.

Co-learning is one of the semi-supervised learning strategies. It allows two classifiers to learn from each other and finally they are integrated to achieve better performance. Its algorithm is shown as follow[17]:

Algorithm 1: Co-learning algorithm

```
Input: labeled data ${(x_i, y_i)}_{i=1}^{l}$, unlabeled data ${x_j}_{j=l+1}^{l+u}$, a
    learning speed $k$. Each instance has two views
    $x_i = [x_i^{(1)}, x_i^{(2)}]$.
1. Initially let the training sample be
    $L_1 = L_2 = \{(x_1, y_1), \cdots, (x_l, y_l)\}$.
2. Repeat until unlabeled data is used up:
3. Train a view-1 classifier $f^{(1)}$ from $L_1$, and a view-2
   classifier $f^{(2)}$ from $L_2$.
4. Classify the remaining unlabeled data with $f^{(1)}$ and
   $f^{(2)}$ separately.
5. Add $f^{(1)}$'s top $k$ most confident predictions
   $(x, f^{(1)}(x))$ to $L_2$.
    Add $f^{(2)}$'s top $k$ most confident predictions
   $(x, f^{(2)}(x))$ to $L_1$.
    Remove these from the unlabeled data.
```

2.2. Local 2bit Binary Patterns(2bitBP) Feature

2bitBP feature outputs 2bit codes in terms of image's intensity information, so one whole 2bitBP feature can encode 4 different codes in decimal representation. The theory of generating codes is depicted in Fig. 1. We choose this feature because of its consistency to luminance change. Detailed explanation can be found in [16].

2.3. Random Fern Classifier

We adopt this classifier because it is simple, computational fast and easy to implement incremental learning. In our tracking case, a random fern classifier consists of several ferns and each fern contains several 2bitBP features. When it comes to training phase, labeled samples are fed into all ferns to update the distributions for the corresponding classes. So at the end of the training we will finally obtain distributions over possible fern outputs for each class. When a new unlabeled sample comes, it selects bins of distributions for each fern according to 2bitBP code and these bins are then combined assuming independence between distributions. Finally, this sample is classified to the category of which the posterior is the biggest, see [18][19] and *http://cvlab.epfl.ch/alumni/oezuysal/ferns.html* for detail.

3. ALGORITHM IN DETAIL

Tracking is considered as a classification problem in our algorithm. The features include RGB histogram, HoG and 2BitBP while classifiers are SVM, SVM and random fern accordingly. The unlabeled samples are collected in each frame and they are exploited to enhance classifiers' performance. An overview of our algorithm is showed below:

Algorithm 2: Overview of our tracking algorithm

```
1. Choose object to be tracked manually.
2. Use mean-shift tracker to collect enough labeled samples
   ${(x_i, y_i)}_{i=1}^{l}$ in the first $S$ frames. Sample sets belonging
   to three classifiers $L^{(RGB)}$, $L^{(HoG)}$ and $L^{(2bitBP)}$ are all
   equal to $\{(x_1, y_1), \cdots, (x_l, y_l)\}$ at this time.
3. Initialize three classifiers $f^{(RGB)}$, $f^{(HoG)}$, $f^{(2bitBP)}$
4. Calculate classifiers' weights $\omega^{(RGB)}$, $\omega^{(HoG)}$, $\omega^{(2bitBP)}$.
For frame=S+1 to N      //N is the total frame number
    5. Collect unlabeled samples ${x_j}_{j=l+1}^{l+u}$, classify them by
       $f^{(RGB)}$, $f^{(HoG)}$, $f^{(2bitBP)}$ separately.
    6. Use constraints to revise classification results.
    7. Construct three confidence maps $M^{(RGB)}$, $M^{(HoG)}$,
       $M^{(2bitBP)}$ for each classifier.
    8. Merge $M^{(RGB)}$, $M^{(HoG)}$, $M^{(2bitBP)}$ to a final confidence
       map based on $\omega^{(RGB)}$, $\omega^{(HoG)}$ and $\omega^{(2bitBP)}$.
    9. Take the peak of confidence map as object location.
    10. Update classifiers:
        Add $f^{(RGB)}$'s top $k$ most confident predictions
        $(x_p, f^{(RGB)}(x_p))$ $p = 1, 2 \cdots k$, to $L^{(HoG)}$, $L^{(2bitBP)}$.
        Add $f^{(HoG)}$ 's top $k$ most confident predictions
        $(x_p, f^{(HoG)}(x_p))$ $p = 1,2\cdots k$, to $L^{(RGB)}$, $L^{(2bitBP)}$.
        Add $f^{(2bitBP)}$ 's top $k$ most confident predictions
        $(x_p, f^{(2bitBP)}(x_p))$ $p = 1,2\cdots k$, to $L^{(RGB)}$, $L^{(HoG)}$.
    11. Update weights $\omega^{(RGB)}$, $\omega^{(HoG)}$, $\omega^{(2bitBP)}$.
    12. Prune out-of-date samples before $T$ frame.
end
```

3.1. Initialization

The target of this step is to collect enough labeled samples for later learning. Our algorithm requires user to mark the target object in the first frame. Then this instance is considered as a positive sample. In the first S frames, a temporary existing simple tracker, like mean-shift, is utilized to get more positive samples. Since S is very small, it can perform stable result during such a short time.

The instances marked as moving object in the first S frames by user and mean-shift tracker are believed to be positive samples. Negative samples can be obtained by selecting instances which partially overlap the object, see Fig. 2. Besides, more additional labeled samples should be generated by making luminance change or adding some noise to the original collected samples. This measure can obviously improve tracking robustness.

3.2. Exploiting Unlabeled Samples

When a new frame arrives, many unlabeled samples will be gathered in searching region by using sliding window technique. The feature vector of sample X is denoted as $X = [x^{(RGB)}, x^{(HoG)}, x^{(2bitBP)}]$. Then three classifiers $f^{(RGB)}$, $f^{(HoG)}$, $f^{(2bitBP)}$ are allowed to allocate labels to these unlabeled samples individually. For building individual confidence map, confidence level of classification for each sample is needed. To what degree a sample classified by SVM should be trusted is measured by the distance from it to the support hyperplane in feature space, namely the farther away one sample from support hyperplane, the more confident of its label prediction. And the confidence rate of random fern classifier can be evaluated by the height margin between positive and negative bins in posterior and larger margin means higher confidence level.

3.3. Applying Constraints

We have employed certain constraints to prevent classification errors. One common sense is that the target in current frame should not too far from its position in the previous frame. Besides, the appearance of object should not change too much in two adjacent frames.

In fact, these misclassified samples amended by constraints are sources for causing drifting. Once mis-tracking occurs in one frame, error will be probably accumulated with classifiers' update again and again. From this perspective, though these constraints are very simple, they can efficiently avoid drifting problem in our tracking algorithm.

3.4. Locating the Object

After acquiring three individual confidence maps, the next task is to combine them into a final one. This is done by using weight sum principle, so the key is to define weights of three classifiers. One straight approach is to let the weight be inversely proportional to classifier's classification error rate, thus we can obtain:

$$\omega^{(RGB)} = \frac{1-\varepsilon^{(RGB)}}{3-(\varepsilon^{(RGB)}+\varepsilon^{(HoG)}+\varepsilon^{(2bitBP)})} \quad (1)$$

$$\omega^{(HoG)} = \frac{1-\varepsilon^{(HoG)}}{3-(\varepsilon^{(RGB)}+\varepsilon^{(HoG)}+\varepsilon^{(2bitBP)})} \quad (2)$$

$$\omega^{(2bitBP)} = \frac{1-\varepsilon^{(2bitBP)}}{3-(\varepsilon^{(RGB)}+\varepsilon^{(HoG)}+\varepsilon^{(2bitBP)})} \quad (3)$$

$\varepsilon^{(\gamma)}$ ($\gamma = RGB, HoG, 2bitBP$) denotes the classifier's error rate with feature γ . Then the final integrated confidence map can be calculated. The peak of integrated confidence map is regarded as tracking target and we may shift tracking box on the corresponding sample, as depicted in Fig. 3.

3.5. Updating Classifiers

Collaborative learning is introduced to update each classifier, i.e., one classifier's most confident samples are delivered to the other two for updating them. Note that in [1] the sample corresponding to the peak in integrated confidence map is regarded as positive and K sub-peaks are deemed as negative. However, we believe it is not a good idea that there must be only "one" positive sample generated in each frame. If confidence levels of positive samples are really low while those of negative are high, it will be better only to select these negative samples for updating, because positive samples having low confidence are probably classified incorrectly. Another case may appear that several positive samples possess very high confidence simultaneously, then it doesn't matter to use all of them to update classifiers. We do this because these positive samples are mostly neighbors of real target object and they are nearly the same with tracking target. In one word, in each frame the number of generated positive samples is not limit to one, which just depends on samples' confidence level.

As mentioned above, $f^{(RGB)}$ and $f^{(HoG)}$ are SVMs indeed. But in our case, the size of positive samples is far smaller than that of negative samples, so it is necessary to assign different penalty coefficients to positive and negative samples when training SVM classifiers. Suppose there are $L_P^{(\gamma)}$ positive and $L_N^{(\gamma)}$ ($\gamma = RGB, HoG$) negative labeled samples in $f^{(\gamma)}$'s current dataset, the penalty coefficients are denoted as $C_P^{(\gamma)}$ and $C_N^{(\gamma)}$, then $C_P^{(\gamma)}$ and $C_N^{(\gamma)}$ should satisfy:

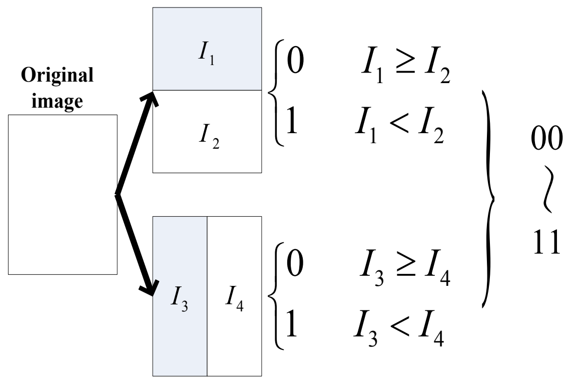
Original image
I_1
I_2
I_3
I_4
$\begin{cases}0 & I_1 \ge I_2\\ 1 & I_1 < I_2\end{cases}$
$\begin{cases}0 & I_3 \ge I_4\\ 1 & I_3 < I_4\end{cases}$
00 ~ 11

Figure 1. Schematic diagram of 2bitBP feature

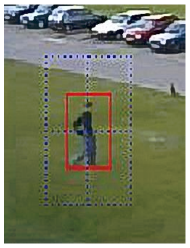

Figure 2. Positive & negative samples
(real rectangular: positive; dot rectangulars: negative)

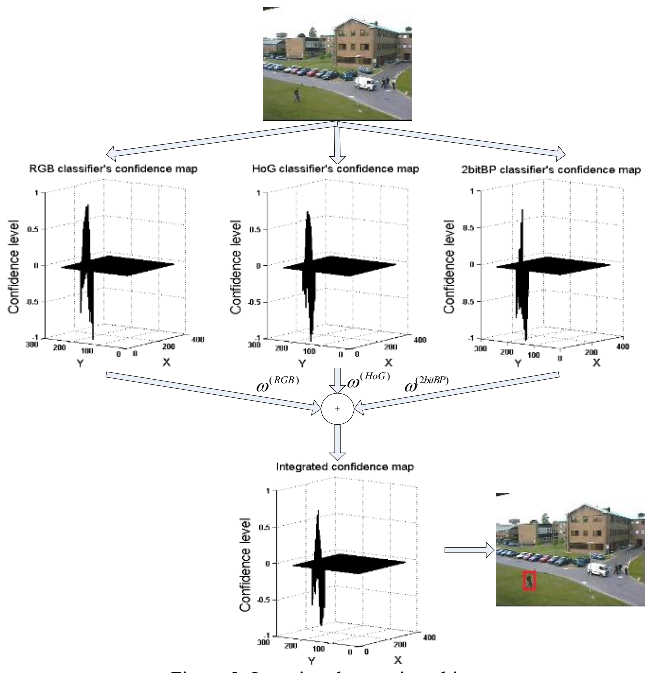
RGB classifier's confidence map
HoG classifier's confidence map
2bitBP classifier's confidence map
Confidence level
$\omega^{(RGB)}$
$\omega^{(HoG)}$
$\omega^{(2bitBP)}$
Integrated confidence map

Figure 3. Locating the moving object

$$\frac{C_P^{(\gamma)}}{C_N^{(\gamma)}} = \frac{L_N^{(\gamma)}}{L_P^{(\gamma)}} \tag{4}$$

In later discussion, we will omit superscript γ for simplicity, and then the form of SVM is derived as:

$$\begin{aligned}&\min_{\omega,b,\xi} \quad \frac{1}{2}\|\omega\|^2 + C_P\sum\xi_P + C_N\sum\zeta_N\\ &s.t. \quad y_i(\omega^T x_i + b) \ge 1-\xi_i \quad (i=1,2,\cdots L_P+L_N)\\ &\qquad\ \xi_i \ge 0\end{aligned} \tag{5}$$

Where ξ_i are slack variables. The dual form of (5) is straightforward:

$$\begin{aligned}&\min_{\alpha} \quad \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j K(x_i \bullet x_j) - \sum_i \alpha_i\\ &s.t. \quad \sum_i y_i\alpha_i = 0\\ &\qquad 0 \le \alpha_i \le C_P, \quad y_i = positive\\ &\qquad 0 \le \alpha_i \le C_N, \quad y_i = negative\end{aligned} \tag{6}$$

Updating random fern classifier is much easier. After samples are labeled, they are sent into all ferns to modify distributions for the corresponding class on the basis of original distributions. That's why we claim fern classifier is suitable for incremental learning in section 2.3.

After being updated, the classifiers' weights should also be modified according to formula (1)～(3).

3.6. Trimming Samples

The capacity of sample dataset will grow very quickly if all samples in the history are preserved, so part of them should be removed. An intuition is that the samples collected in nearest T frames are more significant, so all the samples before T frames are trimmed off. By doing this, classifiers can learn object's latest appearance in time, which is better for precise tracking and adaptivity improvement.

3.7. Summary

From section 3.1 to 3.6, we have explained some key steps in our algorithm. The whole tracking procedure is shown in Fig. 4.

4. EXPERIMENTAL RESULTS

Our proposed approach is implemented in MATLAB and tested with different types of sequences. LibSVM[20] is employed to train SVM classifier in practical. Parameter S is set to 10 and T equals to 50.

4.1. Tracking Pedestrian

This public sequence comes from PET2001 dataset and we extracted the period during which the object is scale consistent. Two popular algorithms: mean-shift, co-tracker in [1], and our method are tested on this sequence separately. Fig. 7 shows some key frames.

Through comparison we can clearly see that both mean-shift and co-tracker are not capable of avoiding drifting, while our method performs better result. Several possible reasons are analyzed below:

(I) We notice that if mean-shift tracker is used, the tracking box is taken away by the lamppost when target is walking pass it(see frame 133～181 of 1st row in Fig. 7).It suggests that tracker is easily confused if only one single feature is employed(like mean-shift), so more proper features are better off being used, just like what we have done in this paper.

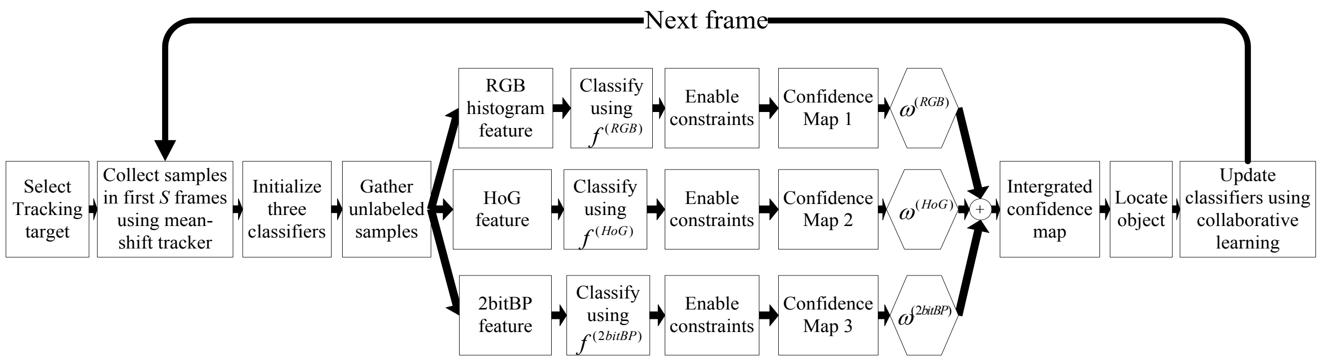

Figure 4. The flowchart of tracking procedure

(II) When co-tracker algorithm is adopted, we see that drifting appears when the pedestrian is walking near the boundary between grass and road. This is perhaps because abrupt background change will cause contradiction between the two classifiers, so one more classifier introduced by us is necessary to deal with such situation. In order to visualize this point, confidence maps of three classifiers in 133rd frame are presented in Fig. 5. In this frame, the target is occluded by a lamppost, which leads to contradictory between $f^{(RGB)}$ and $f^{(HoG)}$ for determining the most confident samples(positive and negative). At this time, it is the addition of the third classifier $f^{(2bitBP)}$ that helps a lot to decide where is the target and which samples are background indeed.

Just because of these reasons, the algorithm in this paper achieves better performance. Integrated confidence level's curve is shown in Fig. 6.

We can see that during frame 120～140, when pedestrian is walking behind the lamppost, the confidence level drops significantly. However, our algorithm is still able to track the pedestrian correctly.

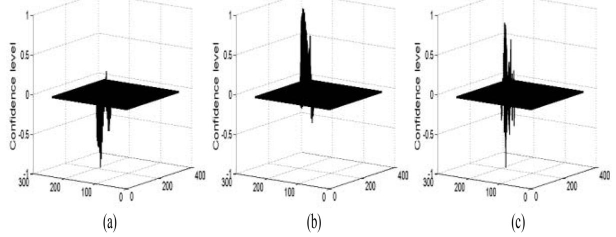

Figure 5. Confidence maps of three classifiers in 133rd frame. (a) represents $f^{(RGB)}$, (b) denotes $f^{(HoG)}$ and (c) belongs to $f^{(2bitBP)}$.

4.2. Occlusion Handling

We have tracked a cup which is occluded by a box. Occlusion is happening as shown in Fig. 8(frame 827～938) We see that tracking box sways slightly when the cup is partially covered by a box. However, if the box is removed, the tracking box won't be taken away and target is still tracked accurately.

4.3. Tracking face

Our algorithm is also suitable for multi-view face tracking. Fig. 9 shows the tracking result on public sequence *Foreman* (Available at http://trace.eas.asu.edu/yuv/index.html). This sequence contains severe expression change and different head gestures. Though target's appearance changes a lot, robust tracking can be also realized.

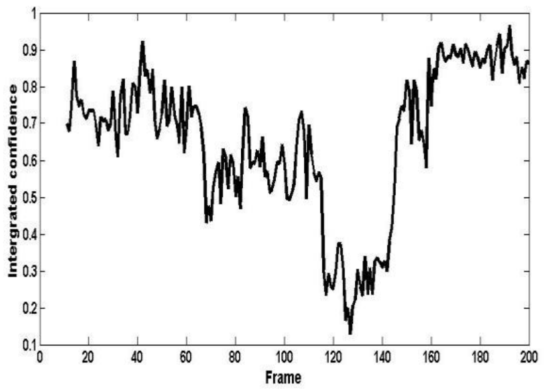

Figure 6. Curve of integrated confidence

5. CONCLUSION

In this paper, we have extended the method in [1] by incorporating 2bitBP feature and random fern classifier. Three classifiers are updated in a collaborative fashion during tracking process to learn the changes of environment and tracking target. Besides, some measures have been taken to improve tracking robustness and adaptivity such as implementation of constraints, novel sample selection manner and so on. Our method has been tested on some typical sequences and performed impressive results.

However, this algorithm cannot handle object scale change, but if searching boxes are set in different scales when sliding window technique is executed, target in scale change can be easily resolved. In our opinion, bringing more machine learning approaches into object tracking will be a trend in the future.

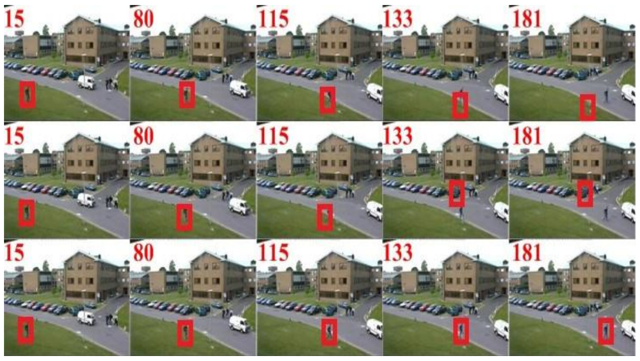

Figure 7. Tracking results comparison
(1st row: mean-shift, 2nd row: co-tracker in [1], 3rd row: our method)

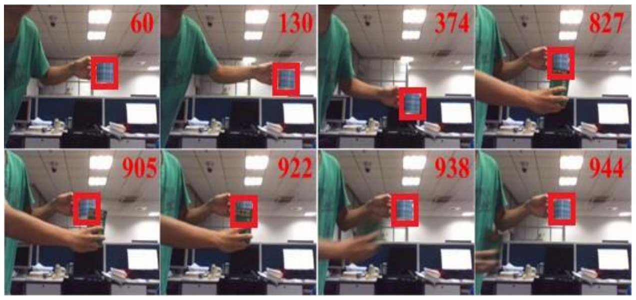

Figure 8. Tracking occluded cup

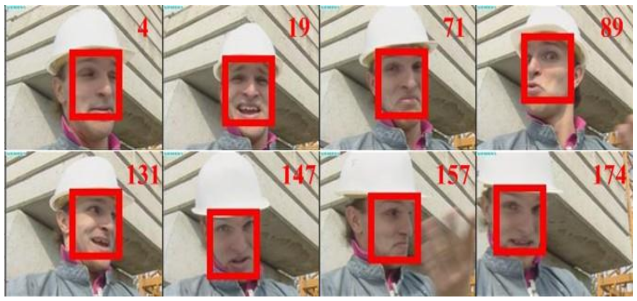

Figure 9. Tracking face

References

[1] F. Tang, S. Brennan, Q. Zhao, and H. Tao, "Co-tracking using semi-supervised support vector mach-ines", *Int. Conf. Computer Vision*, pp. 1–8, Oct. 2007.

[2] D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an application to Stereo Vision" *Proceedings of Imaging Understanding Workshop*, pp. 121-130, Apr. 1981.

[3] S. C. Park, S. H. Lim, B. K. Sin and S. W. Lee, "Tracking non-rigid objects using probabilistic Hausdorff distance matching", *Pattern Recognition*, vol. 38, pp. 2373-2384, Dec. 2005.

[4] H. T. Nguyen, M. Worring and R. Van den Boomagaard, "Occlusion robust adaptive template tracking", *Int. Conf. on Computer Vision*, vol. 1,pp. 678-683, Jul. 2001.

[5] A. Senior, "Tracking people with probabilistic appearance models", *3rd International workshop on Performance Evaluation of Tracking and Surveillance Systems*, pp. 48-55, 2002.

[6] D. Comaniciu, V. Ramesh and P. Meer, "Kernel-based object tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 564-575, May. 2003.

[7] M. Isard and A. Blake, "CONDENSATION- Conditional density propagation for visual tracking", *Int. Journal of Computer Vision*, vol. 29, pp. 5-28, Mar. 1998.

[8] S. Avidan, "Support vector tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 26, pp. 1064–1072, Aug. 2004.

[9] Z. Kalal, K. Mikolajczyk and J. Matas, "Face-TLD: Tracking-Learning-Detection applied to faces", *Int. Conf. Image Processing*, pp. 3789-3792, Sept. 2010.

[10] H.Grabner and H.Bischof, "On-line boosting and vision", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 260–267, Jun. 2006.

[11] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking", *Proc. European Conference on Computer Vision*, vol. 5302. pp. 234-247, Oct. 2008.

[12] Y. F. Zha, Y. Yuan and D. Y. Bi, "Graph-based transductive learning for robust visual tracking", *Pattern Recognition*, vol.43, pp. 187-196, Jan. 2010.

[13] Q. Yu, T. Dinh and G. Medioni, "Online tracking and reacquisition using co-trained generative and dis-criminative trackers", *Proc. European Conference on Computer Vision*, vol. 5303, pp. 678-691, Oct. 2008.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, Jun. 2005.

[15] Z. Kalal, J. Matas and K. Mikolajczyk, "P-N learning: Bootstrapping binary classifiers by structural constraints", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 49–56, Jun. 2010.

[16] Z. Kalal, J. Matas and K. Mikolajczyk, "Online learning of robust object detectors during unstable tracking", *IEEE Int. Conf. Computer Vision Workshops*, pp. 1417-1424, Sept. 2009.

[17] X. Zhu and A. Goldberg, "Introduction to semi-supervised learning", *Morgan & Claypool Publishers*, 2009.

[18] M. Ozuysal, P. Fua, and V. Lepetit, "Fast keypoint recognition in ten lines of code", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1-8, Jun. 2007.

[19] M. Ozuysal, M. Calonder, V. Lepetit and P. Fua, "Fast Keypoint Recognition using Random Ferns", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 448–461, Mar. 2009.

[20] C. C. Chang and C. J. Lin, "LIBSVM : a library for support vector machines", *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu. edu.tw/~cjlin /libsvm.