Robust visual tracking based on Informative random fern

Hao Dong[1, a], Rui Wang[1, b]

[1]School of Instrumentation Science and Opto-electronics Engineering, Beihang University, Beijing 100191, China;

[a]qianxin_dh@163.com, [b]wangr@buaa.edu.cn

Keywords: Visual tracking, IRF-TLD, Gaussian projection, Real time

Abstract. In this paper, a novel visual tracking algorithm named as Informative random fern - Tracking Learning Detection (IRF-TLD) has been proposed. Instead of a binary comparison in the standard random fern of TLD, we use the real value feature and Gaussian random projection to acquire the advantages of high accuracy and low memory requirement. Experimental results on challenging sequences have demonstrated the superior performance of our IRF-TLD when compared with several state-of-the-art tracking algorithms.

1. Introduction

Visual tracking is one of the most important problems in computer vision. It is the basis for many applications such as surveillance, human computer interaction and action recognition, etc. Many methods have been proposed for visual tracking over the past few decades. Generally speaking, most trackers can be divided into two categories: generative models and discriminative models. Generative models [1] are typically formulated as searching the most similar image region with minimal reconstruction error. Owing to the fact that they concern only about the appearance of the object, the generative models often fail in cluttered background. For discriminative models [2], tracking is treated as a binary classification task that finds the decision boundary between the target and the background. Compared with generative models, discriminative models are usually more resistant to cluttered background since they explicitly sample image patch from the background as negative example to train the classifier.

Kalal et.al [3] proposed a novel approach called Tracking Learning Detection(TLD), in which tracking and detection are independent processes that exchange information via learning. Random fern [4] classifier is one important component of the cascade detector in TLD and shows excellent performance. However, there exist some potential problems with it. First, the comparison of each pixel pair produces only two outputs, 0 or 1, leading to lots of information loss. In addition, the random fern classifier in TLD requires enormous memory, having an exponential relationship with the number of pixel pairs in a fern. To address the issues, we extend the TLD based on informative random fern which produces the real value feature for a fern based on subtraction and Gaussian projection.

The rest of this paper is organized as follows. In Section 2, the introduction of random fern is introduced. The proposed method IRF-TLD is presented in Section 3. Section 4 shows the experimental results, followed by conclusion in Section 5.

2. Preliminaries

In random fern, the simple intensity comparisons between pixel pairs are chosen as the binary features. Let $f_i, i = 1,..,N$ denotes the binary feature that extracted from an image patch which to classify. The class c for this image can be described by

$$c = \arg\max_{c \in C} p(c \mid f_1, f_2, ..., f_N) \tag{1}$$

Here, C is the set of all classes. Using Bayes' formula, the posterior can be written as

$$p(c \mid f_1, f_2, ..., f_N) = \frac{p(f_1, f_2, ..., f_N \mid c)p(c)}{p(f_1, f_2, ..., f_N)} \tag{2}$$

We take the denominator as a constant and assume the probability $p(c)$ is uniform, then (1) is equal to

$$c = \arg\max_{c \in C} p(f_1, f_2, ..., f_N \mid c) \tag{3}$$

Ozuysal et al. [4] proposed to divide the features into several groups, and assumed the different groups are independent of each other. Formally,

$$p(f_1, f_2, ..., f_N \mid c) = \prod_{i=1}^{N/S} p(F_i \mid c) \tag{4}$$

Where S is the number of pixel pairs in each fern and F_i is a group of features, named as a fern. $T = N/S$ is the total number of ferns. In practice, S cannot be too small, thus the memory occupation is very enormous.

3. IRF-TLD algorithm

3.1 IRF-TLD framework

Our whole IRF-TLD tracking approach is summarized in Fig.1. It inherits the framework of TLD which decomposes the long-term tracking task into tracking, detection and learning. The target is followed by a tracker from frame to frame and its motion is estimated using the Lucas-Kanade tracker extended with failure detection. The task of learning is to initialize the cascade detector in the first frame and update it in run-time using the P-N experts. In original TLD, the cascade detector, which is responsible for selecting the most possible target candidate in each frame, consisted of three stages: (i) patch variance: this stage can rejects those patches with gray-value variance smaller than 50 percent of the variance of the target patch; (ii) random fern: it performs a quantity of pixel comparisons on a patch resulting in a binary code, which indexes to an array of posteriors. (iii) nearest neighbor: it is the last stage to divided each candidate patch into target object or background by appearance using Normalized Correlation Coefficient.

In our IRF-TLD, the tracker and learning methods from TLD are adopted. Meanwhile, some improvements are made in the cascade detector. Instead of the binary comparison in the original random fern of TLD, we introduce the informative random fern classifier to improve the robustness of the detector. The more informative real value from the subtraction is used in our method. Moreover, a random projection is utilized to map the value of each fern derived from feature value to a parametric distribution, specifically, Gaussian distribution, in which the classification is done. In the following, the proposed IRF classifier will be described in detail from three steps: feature formation, classification with probability and online update.

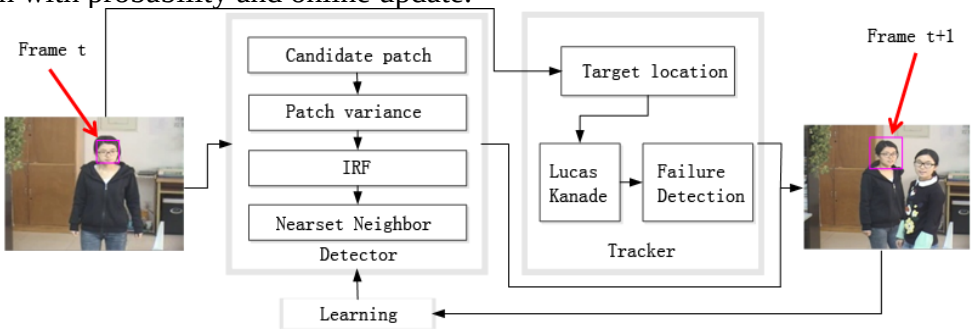

Fig. 1 Framework of IRF-TLD tracking algorithm

3.2 Feature Formation

We adapt the real value feature from [5], i.e., the real value feature $f_{i,j}$ described in Eq.(5) is extracted from pixel pair j of fern i:

$$f_{i,j} = I(d_1(i,j)) - I(d_2(i,j)), \tag{5}$$

Where $I(d)$ represents the intensity of an image patch I at d. $d_1(i,j)$ and $d_2(i,j)$ denote the

coordinates of the randomly generated pixel pair j of fern i. Obviously, the real value feature can preserve more information about the intensity difference between two pixels because of $f_{i,j} \in \mathbb{R}$ instead of $f_{i,j} \in \{0,1\}$.

Since the feature $f_{i,j}$ is a real value, it's necessary to "encode" all real values in each fern into a single real value to simplify the subsequent classification. A theoretical basis for this idea has been stated by Johnson-Lindenstrauss(JL) lemma [6] that with high probability the distances between the points in the high-dimensional space are preserved if they are projected onto a randomly selected low-dimensional subspace. Besides, the literature [6] also proved that for k-sparse data (e.g, image and audio signal), the random matrix such as Gaussian matrix satisfying the JL lemma holds true for the restricted isometry property in compressive sensing. Therefore, we use the Gaussian matrix to facilitate efficient projection from feature values of different pixel pairs into a single real value in this paper. Formally,

$$F_i = \sum_{j=1}^{S} r_j f_{i,j} \tag{6}$$

Where $r_j \sim N(0,1)$ is a real value generated randomly according to a Gaussian distribution.

Besides, comparing the proposed IRF with the standard random ferns method, we can find that the IRF has the advantages of requiring a constant and much lower memory from the following analysis. Assuming that the number of classes is $\gamma = 2$ (foreground and background) and the real value feature is stored in a single precision type which occupies 4 Bytes. Then the memory requirement is $MEM_{Our} = T \times \gamma \times 4$. While in the standard random ferns method used in TLD, a specific binary code is stored in an integral type which occupies 4 Bytes. The memory requirement is $MEM_{TLD} = 2^S \times T \times \gamma \times 4$. It's clear that the standard random fern method in TLD needs memory 2^S times more than the proposed IRF method.

3.3 Classification with probability:

In IRF, the output F_i is calculated as a single real value produced randomly on the basis of Gaussian distribution. For simplicity, we model the probability $p(F_i \mid c)$ as a Gaussian distribution with parameters $\left(\mu_i^c, \sigma_i^c\right)$ for fern i of class c. Whereupon, the discriminative function is

$$\begin{aligned} H(F) &= \log\left(\frac{\prod_{i=1}^{T} p(F_i \mid c=1)p(c=1)}{\prod_{i=1}^{T} p(F_i \mid c=0)p(c=0)}\right) \\ &= \sum_{i=1}^{T} \log(p(F_i \mid c=1)) - \sum_{i=1}^{T} \log(p(F_i \mid c=0)) \end{aligned} \tag{7}$$

Where we assume uniform prior $p(c=1) = p(c=0)$, $c \in \{0,1\}$ is a binary variable which represents the sample label and $F = \{F_1, F_2, ..., F_T\}$ is a set containing the value of all ferns for an image patch.

The IRF classifies the patch as the target if the corresponding value $H(F)$ is larger than zero.

3.4 Online Update:

To integrate our IRF feature that the value of each fern is modeled as a Gaussian distribution with parameter $\left(\mu_i^c, \sigma_i^c\right)$ to the target model, we simplify the update of the classifier as a parameter update:

$$\begin{aligned} \mu_i^c &\leftarrow \lambda \mu_i^c + (1-\lambda)\mu_i^{c,new} \\ \sigma_i^c &\leftarrow \sqrt{\lambda(\sigma_i^c)^2 + (1-\lambda)(\sigma_i^{c,new})^2 + \lambda(1-\lambda)(\mu_i^c - \mu_i^{c,new})^2} \end{aligned} \tag{8}$$

Where λ is the learning rate, $\mu_i^{c,new} = E[F_i \mid c]$ and $\sigma_i^{c,new} = \sqrt{E[(F_i \mid c)^2] - (E[F_i \mid c])^2}$ are estimated from the training samples that are generated by P-N experts at current frame.

4. Experiments

Our ITTS is implemented in C++, which runs at 25 frames per second on an Intel Dual-Core 3.30GHz CPU with 4G RAM. Three state-of-the-art algorithms on 6 fully-annotated video sequences included TLD [3], OAB [7] and CT [2] are used to validate the performance of our IRF-TLD

algorithm. All of these algorithms are evaluated in the one-pass evaluation(OPE) [8], and these sequences with the corresponding ground truth files and the compared code library are available on the website: http://visual-tracking.net. In all the experiments, the total number of ferns is set to $T = 50$, the number of pixel pairs in a fern is decided as $S = 4$, and the learning rate λ is selected as 0.85.

4.1 Evaluation Metric

We use the precision plots and success plots [8] to evaluate the robustness of tracking algorithms quantitatively. The precision plot shows the percentage of frames whose estimated center locations are within the given threshold distance of the ground truth. To compare the performances of different algorithms, the score for the threshold equal to 20 pixels is used to be the representative precision score. Meanwhile, the success plot is based on the overlap ratio that is $OS = \text{Area}(b_t \cap b_a)/\text{Area}(b_t \cup b_a)$, where b_t is the tracked target box and b_a denotes the ground truth box. The success plot shows the ratios of frames with $OS > t_0$ throughout all threshold $t_0 \in [0,1]$. The area under curve(AUC) of each success plots serves as the first measure to rank the tracking algorithms in the following.

4.2 Result and Analysis

The overall performance of the 4 tracking algorithms based on success plots and precision plots are illustrated in Fig.2. According to the experimental results, our algorithm achieves outstanding performances in both the metric overlap and center location error: in the success plot, it achieves an AUC score of 0.581 and ranks 1st. Moreover, our IRF-TLD algorithm outperforms TLD by 6.8%. Meanwhile, the overall precision of our IRF-TLD at 79.3% is still the highest among all algorithms, yet beating TLD by 1.1%.

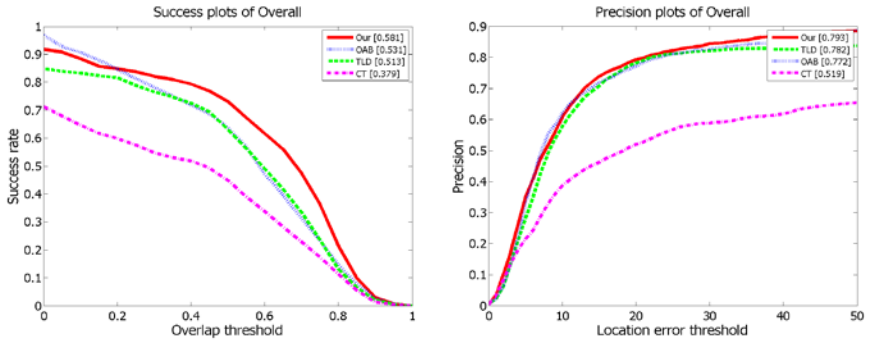

Fig. 2 The overall performance of the 4 tracking algorithms on all video sequences

To further analyze the performance of IRF-TLD, the AUC scores and precision scores for each sequence are also shown in Table 1. Some sampled results on sequences are illustrated in Fig.3. From Table 1, we can observe that IRF-TLD performs best on 4 out of 6 sequences (the ***italic*** fonts indicate the best performance). Note that there exist many challenging factors in these videos that IRF-TLD achieves favorable results. For instance, the sequences *faceocc2*, *carscale*, and *sylverster* have the attributes of scale variation and (in-)out-of plane rotation, in which *faceocc2* and *carscale* also have occlusion attribute, thereby making them far more challenging. Notwithstanding, IRF-TLD performs persistently well from beginning to end.

Table 1 The AUC/Precision scores on each sequence

Sequences	Our	TLD	OAB	CT
Faceocc2	***0.619***/0.792	0.611/0.856	0.593/0.708	0.602/0.681
Sylvester	***0.676***/0.946	0.666/0.949	0.557/0.741	0.659/0.901
Carscale	***0.575***/0.718	0.452/0.853	0.398/0.663	0.433/0.718
Skating	0.366/0.495	0.190/0.318	***0.394***/0.688	0.086/0.090
Doll	0.561/0.918	***0.566***/0.983	0.533/0.874	0.455/0.684
Deer	***0.690***/0.915	0.590/0.732	0.640/0.958	0.039/0.042

IRF-TLD not merely inherits the original tracking framework of TLD, the superiority of our IRF-TLD algorithm compared with TLD mainly lies in that: its cascade detector performance is further improved by combining the IRF classifier. In addition, IRF-TLD produces the real value for a fern based on subtraction and Gaussian projection, leading to a more informative result than the

binary feature used in the TLD. Hence, the maintaining of the diversity of real value features enables IRF-TLD to practice excellently in the presence of significant drastic appearance changes.

5 Conclusion

In this paper, we have proposed a novel tracking method based on TLD and Informative random fern. The proposed method has advantages of high accuracy and low memory requirement, thus is very appropriate for embedded systems. Experimental results show that it performs better than some other methods on most video sequences.

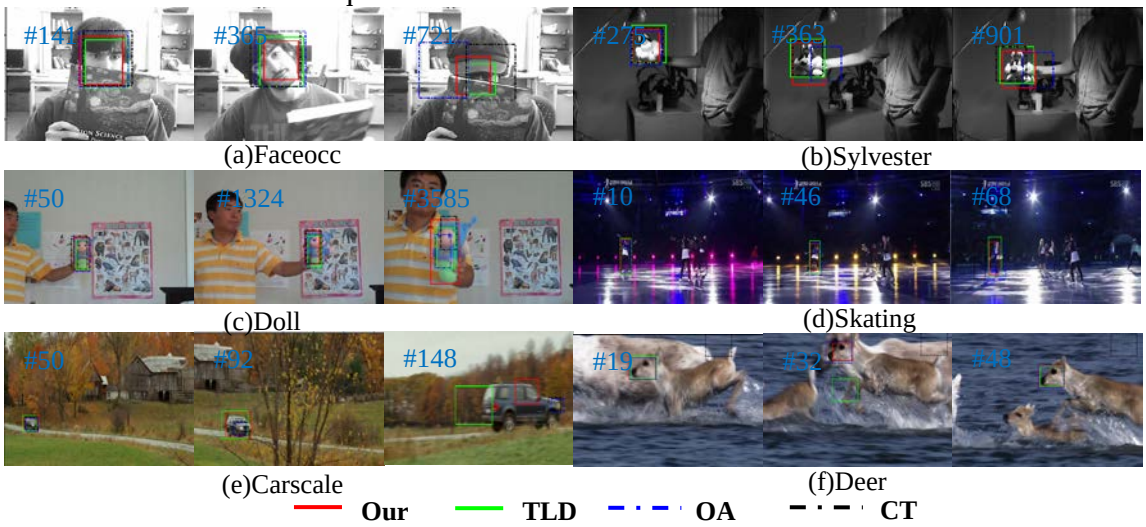

Fig. 3 Screenshots from some of some sampled tracking results.

Acknowledgement

This work was partially supported by National Natural Science Foundation of China(60974108).

References

[1]. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H. Incremental Learning for Robust Visual Tracking[J]. International Journal of Computer Vision, 2008, 77(1-3): 125-141.

[2]. Zhang, K., Zhang, L., Yang, M.-H. Real-time compressive tracking[C]. European Conference on Computer Vision. Italy, 2012, pp. 864-877.

[3]. Kalal, Z., Mikolajczyk, K., Matas, J. Tracking-learning-detection[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 34(7): 1409-1422.

[4]. Ozuysal, M., Fua, P., Lepetit, V. Fast keypoint recognition in ten lines of code[J]. IEEE Conference on Computer Vision and Pattern Recognition. USA, 2007, pp. 1-8.

[5]. Zhang, J., Liu, K., Cheng, F., Li, Y. Visual tracking with randomly projected ferns[J]. Signal Processing: Image Communication, 2014, 29(9):987-997.

[6]. Achlioptas, D. Database-friendly random projections: Johnson-Lindenstrauss with binary coins[J]. Journal of computer and System Sciences, 2003, 66(4), 671-687.

[7]. Grabner, H., Grabner, M., Bischof, H. 1 Real-Time Tracking via On-line Boosting[C]. British Machine Vision Conference. UK, 2006, 47-56.

[8]. Wu, Y., Lim, J., Yang, M.-H. A benchmark[C]. IEEE Conference on Computer Vision and Pattern Recognition. USA, 2013, pp. 2411-2418.