VISUAL OBJECT TRACKING VIA RANDOM FERNS BASED CLASSIFICATION

Aniruddha Acharya K. *R. Venkatesh Babu*

Video Analytics Lab, Supercomputer Education and Research Center,
Indian Institute of Science, Bangalore, India

ABSTRACT

Designing a robust algorithm for visual object tracking has been a challenging task since many years. There are trackers in the literature that are reasonably accurate for many tracking scenarios but most of them are computationally expensive. This narrows down their applicability as many tracking applications demand real time response. In this paper, we present a tracker based on random ferns. Tracking is posed as a classification problem and classification is done using ferns. We used ferns as they rely on binary features and are extremely fast at both training and classification as compared to other classification algorithms. Our experiments show that the proposed tracker performs well on some of the most challenging tracking datasets and executes much faster than one of the state-of-the-art trackers, without much difference in tracking accuracy.

Index Terms— Object tracking, Classification, Random Ferns

1. INTRODUCTION

Object tracking in video involves tracking the location of a given object in all frames of a video, given its location in the first frame. The problem has many applications in other high level computer vision tasks like video recognition, traffic monitoring, robot navigation and video indexing and retrieval.

Some of the challenges in object tracking are handling of object pose change, scale change, illumination variation and occlusion. These factors deviate the appearance of object in the first frame of video from the subsequent frames. And that makes localization challenging.

Objects in video can be non-rigid and can change their pose (for example humans). Pose change introduces redistribution of object parts or revealing of parts of the object unseen in the first frame. This makes it difficult to get high accuracy tracking by relying on the first frame appearance in a naive method.

Objects in a video may appear to expand or contract. This may either happen because the object has moved close to or away from the camera, or the object has expanded or shrunk, or due to camera zoom. In any case, to effectively track, the scale change must be recognized and the appropriate bounding box must be selected.

Lighting changes are one another complication during tracking. Illumination changes can modify the appearance to a great extent, and hence trackers must be invariant to them or be adaptive to handle them.

Sometimes objects may get occluded by other objects which are not of interest. It is a great challenge to identify the presence of object behind another object, without observing it. Some trackers like the L1 tracker [1] handle this by using trivial templates in L1 minimization framework.

Many of the existing robust trackers are slow in execution and cannot satisfy the real time requirements of tracking applications. The work that is described in this paper aims at designing a tracker that is real time as well as robust. This paper presents a tracker that is comparable to the state-of-the-art trackers in terms of accuracy, and has better execution speed.

In this paper, tracking is posed as a classification problem. A search window is defined in each frame of the video based on the previous frame tracking result. Overlapping image patches of the search window are assumed to belong to one of the 16 patch classes. Depending on the classification result, a likelihood of object map is created, whose weighted centroid is used as the location of the object. The tracker senses scale changes and adapts to appearance and pose changes of the object.

This paper is organized as follows: The related works are given in Section 2. Section 3 gives a brief overview of ferns and the proposed tracker is explained in Section 4. Experiments conducted and the results are presented in Section 5 and are discussed in Section 6. Section 7 concludes the paper.

2. RELATED WORK

Visual object tracking has received strong attention by researchers. It has been in the literature since many years now, but still is no close to being a solved problem.

Early works in object tracking include the mean shift tracker [2] and SSD tracker [3]. The mean shift tracker incrementally seeks the mode of the likelihood distribution of object location and finds the object location in frames of the video. The likelihood distribution is computed using features like color or gray value. The SSD (Sum of squared distances)

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

tracker searches for the object by computing SSD between the object model obtained by the first frame and the candidate object patches in the frames of video. This tracker is not robust to illumination changes as it relies only on the raw patches.

Collins has proposed a mean shift based blob tracking though scale space [4]. The tracker recognizes scale changes in the object and adjusts the width and height of bounding box at every frame by iteratively applying mean shift across spatial and scale spaces.

Mei et al. [1] proposed a tracker based on sparse representation and L1 minimization. A dictionary is built in which the object can be sparsely represented. L1 minimization is performed at every frame for the candidate objects in particle filter framework and the object is localized based on the particle that provides the least reconstruction error.

The performance and speed of this tracker has been improved by Babu et al. [5], [6] by adapting the sparse representation framework by modeling the object with local patches. Further Ashwini et al. [7] extended the this approach by classifying foreground and background patches for reliable tracking.

Jia et al. [8] proposed a tracker which uses the structural information of the object. Object patches are divided into multiple sub-patches and each of the sub-patches are identified in the next frame using L1 minimization framework.

Few machine learning based trackers were proposed in the past. Avidan [9] used weak classifiers for modeling the object and background classes. Babu et al. [10] used extreme learning machines for rapid object background classification for tracking.

Random ferns were first introduced in [11] as a descriptor of patches around keypoints of an image. Ferns were used as an object detector and for matching images by matching the keypoints.

Ferns were used in [12] as the object detector part of an integrated tracker called TLD (Tracking-Learning-Detecting) [13]. A learning strategy called p-n learning was integrated with the Lucas Kanade tracker and fern based object detector.

Enhanced random ferns are used in [14] along with NCC (Normalized Cross Correlation) for tracking. Long term tracking is the core aim of this tracker, and ferns are used only as an object detector, unlike in the proposed tracker wherein ferns are directly used for tracking.

3. RANDOM FERNS

We use random ferns [11] as our classifiers and classify patches in the search window of video frames. Ferns have emerged as a simplification of decision tree classifiers. Ferns are non-hierarchical structures which rely on simple binary tests for distinction between classes. Each fern turns out to be a weak classifier, but the ensemble of hundreds of ferns, each with random binary tests as features, is a highly robust classifier.

N binary features are extracted from each of the patches to be classified. Each binary feature f_i is obtained by comparing a random pixel of the patch with a random threshold as shown in the equation:

$$f_i = \begin{cases} 1, & \text{if } I(x_i, y_i) \leq k_i \\ 0, & \text{otherwise} \end{cases}$$

where,I represents the image and (x_i, y_i) is the random location and k_i is the random threshold used by the i^{th} feature. Unlike decision trees, wherein the binary feature test outputs are hierarchically structured to arrive at the class, ferns combine the binary features using a semi naive Bayesian approach: Let c_i denote the i^{th} class . Then the output class is given by

$$\underset{c_i}{\operatorname{argmax}}\{P(C = c_i | f_1, f_2, f_3, ..., f_N)\}$$

where, C is a random variable representing the class of the patch. Using Bayes formula, we can write

$$P(C = c_i | f_1, f_2, ..., f_N) = \frac{P(f_1, ..., f_N | C = c_i).P(C = c_i)}{P(f_1, f_2, f_3, ..., f_N)}$$

Since each feature is very simple, a large N is required for a good classification result. In that case, representing the joint probability will require storing 2^N entries for each class, which is infeasible. To reduce the memory requirement, sets of M features of size $S = \frac{N}{M}$, randomly taken from the set of all features are assumed to be independent. Each of these sets are called ferns. Thus classification is obtained using the conditional probability:

$$P(f_1, f_2, f_3, ..., f_N | C = c_i) = \prod_{k=1}^{M} P(F_k | C = c_i)$$

where, F_k is the k^{th} fern, represented by an integer of S bits.

4. METHODOLOGY

In the first frame, a search window is defined as a bounding box around the known object bounding box as shown in Figure 1, which includes background pixels surrounding the object. The search window is divided into 16 equally sized non-overlapping blocks as shown in Figure 1. 5×5 overlapping patches of each of the blocks are assumed to belong to a class. Thus in every frame, we have 5×5 patches around the object belonging to one of the 16 classes. As can be observed from Figure 1, 4 out of 16 classes belong to the object and the rest of the classes belong to the background. The reason behind choosing 16 classes instead of only 2 classes corresponding to object and background is that it is experimentally

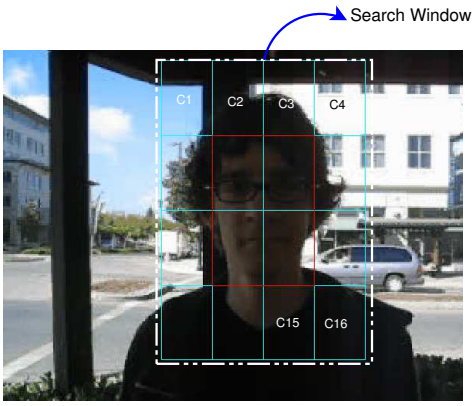

Fig. 1. Search window and the 16 classes

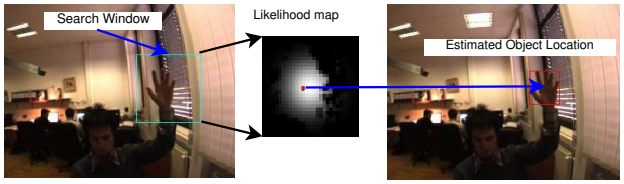

Fig. 2. Left: Search window Center: Gaussian weighted likelihood map with centriod shown in red, Right: Estimated object location

shown in [11] that ferns have better classification accuracy for large number of classes. Moreover, there will be a huge class imbalance in terms of the volume of training data in case of only two classes.

Training phase involves computing $P(F_k|C = c_i)$ for every i and every possible fern value. 5×5 patches are extracted from the initial search window. These patches are used to compute binary features, ferns and fern likelihood of the 16 classes.

During testing, a search window is defined on each frame of the video using the previous frame bounding box. The patches inside the search window are classified using ferns, and a binary likelihood map of object is obtained. The likelihood map is weighed using a Gaussian window centered at the center of the search window to reduce the effect of noisy boundary patches. The centroid of the likelihood map is computed and set as the center of the object. This process is illustrated in Figure 2.

4.1. Handling of scale changes

Object scale may change in the video due to reasons like camera zoom and object movement towards camera. In our tracker, initial object scale is obtained from the scale-space generated using scale-normalized Laplacian of Gaussian filter.

$$LoG(x, y) = -\frac{1}{\pi\sigma^3}[1 - \frac{x^2 + y^2}{2\sigma^2}]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Fig. 3. Left - Plot of Laplacian of Gaussian, Right - LoG response

Figure 3 shows a plot of 2D Laplacian of Gaussian. It has a large negative central lobe and small positive side lobes surrounding it. Laplacian of Gaussians (LoG) are well known to detect the scale of objects in image. The standard deviation σ is a measure of scale of the object. As σ of LoG increases, the width of the central lobe increases. LoG filtering operation can be represented as:

$$Response = LoG(\sigma) * I$$

The LoG whose central lobe exactly fits the object gives the minimum filter output. Hence the right scale(σ) can be found by filtering the image using LoGs with different σ and finding the minimum response. Figure 3 shows LoG response vs σ plot for two images with different scales. It can be observed that σ with the minimum response is higher for the larger scaled image.

We use the binary likelihood map obtained from ferns as the input to LoG filter. A small neighborhood of the scale space around the previous frame scale is used to search for the correct scale in the current frame. Usually the likelihood map is noisy and the estimate of scale is not very accurate. Hence we use the Kalman filter to get a better estimate of the object scale. The width and height of the object bounding box is adjusted according to the scale identified.

4.2. Handling of pose and appearance changes

Object pose and appearance changes are one of the major causes for the difficulty in object tracking. The proposed tracker handles these by adopting an update in the model. The fern model consists of the probabilities $P(F_k|C = c_i)$. As tracking progresses, we include the effect of newly obtained search window patches. Since a wrong update can be disastrous, only a random fraction of the search window patches are used. The training process is re-run by including the new patches to obtain the updated model. To have a limit on the training data, we discard old training samples whenever the training data size reaches a threshold.

5. EXPERIMENTS AND RESULTS

We have implemented the fern based tracker on a 2.30 GHz i5 processor, and evaluated its performance on challenging videos. We have used MATLAB for programming and used the Piotr's toolbox [16] for fern training and testing.

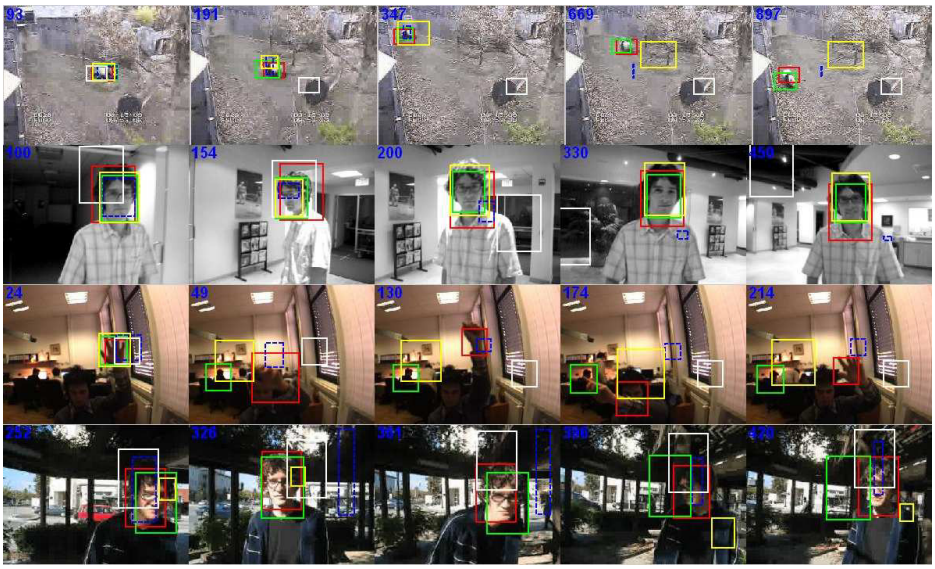

Fig. 4. Output of different trackers: Red-Proposed, white-Mean Shift[4], yellow-TLD[13], blue:L1APG[15], green-VT [8]

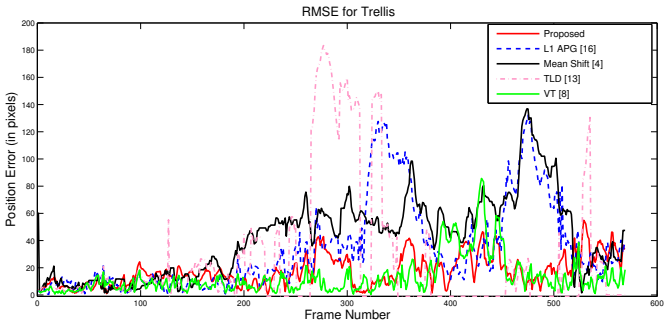

Fig. 5. RMSE vs frame number for Trellis video

We have compared our tracker with the mean shift tracker [4], TLD tracker [13], L1-APG tracker [15] and the visual tracker [8]. Datasets used are panda video, Trellis, hand, and David indoor. These datasets have good amount of pose, lighting, and appearance changes in them.

Figure 4 shows the results of the proposed tracker along with that of the other trackers for different frames of the datasets. Table 1 shows the average root mean squared error (RMSE) of the trackers with respect to ground truth. Figure 5 shows the plot of RMSE vs frame number for the trellis video for different trackers.

6. DISCUSSION

The results in Table 1 show that the proposed tracker is far better than Mean shift, TLD and L1-APG in terms of tracking accuracy. As compared to [8], the tracker accuracy is lesser for Trellis by a small margin and for david by a significant margin. However, our tracker is almost 7 times faster than [8]. Our tracker runs at 8.8 fps whereas [8] runs at 1.27 fps.

Almost 85% of the time spent by our tracker is devoted to model updation using unoptimized MATLAB code, since fern posteriors are re-computed for every update. Time required for fern classification is much less than that required for training.

Table 1. Average RMSE

Sequence	Mean Shift[4]	TLD[13]	L1APG [15]	VT[8]	Proposed
Panda	148.80	55.05	47.06	8.31	5.85
David Indoor	128.60	8.12	59.40	3.61	22.13
Hand	103.44	93.89	66.96	135.79	31.72
Trellis	51.07	66.30	47.41	18.77	19.97

7. CONCLUSIONS AND FUTURE WORK

We have presented a fern based tracker. Tracking is posed as a classification problem and random ferns are used as classifiers. The results show that the tracker has an accuracy much more than many of the well known trackers and an accuracy comparable to a state of the art tracker. The advantage of this tracker is that, it has an execution speed-up of almost 7 as compared to the existing state of the art tracker. Our future work will include coming up with a better update rule and use of location priors to increase the tracker accuracy.

8. REFERENCES

[1] Xue Mei and Haibin Ling, "Robust visual tracking using l_1 minimization," in *Proceedings of 12th IEEE International Conference on Computer Vision*, 2009, pp. 1436–1443.

[2] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2000, vol. 2, pp. 142–149.

[3] Gregory D Hager, Maneesh Dewan, and Charles V Stewart, "Multiple kernel tracking with SSD," in *Proceedings of IEEE Conference on CVPR*, 2004, vol. 1, pp. I–790.

[4] Robert T Collins, "Mean-shift blob tracking through scale space," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003, vol. 2, pp. II–234.

[5] R.Venkatesh Babu, "Real-time robust tracking via sparse representation: A mode-seeking approach," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 3919–3923.

[6] R.Venkatesh Babu and Priti Parate, "Interest points based object tracking via sparse representation," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 2963–2967.

[7] M.J. Ashwini, R.Venkatesh Babu, and K.R. Ramakrishnan, "Context-aware real-time tracking in sparse representation framework," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 2450–2454.

[8] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 1822–1829.

[9] Shai Avidan, "Ensemble tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 2, pp. 261–271, 2007.

[10] R Venkatesh Babu, Sundaram Suresh, and Anamitra Makur, "Online adaptive radial basis function networks for robust object tracking," *Computer Vision and Image Understanding*, vol. 114, no. 3, pp. 297–310, 2010.

[11] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua, "Fast keypoint recognition using random ferns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 448–461, 2010.

[12] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk, "PN learning: Bootstrapping binary classifiers by structural constraints," in *Proceedings of IEEE Conference on CVPR*, 2010, pp. 49–56.

[13] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[14] Wei Quan, Jim X Chen, and Nanyang Yu, "Robust object tracking using enhanced random ferns," *The Visual Computer*, pp. 1–8, 2013.

[15] Chenglong Bao, Yi Wu, Haibin Ling, and Hui Ji, "Real time robust l1 tracker using accelerated proximal gradient approach," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 1830–1837.

[16] Piotr Dollár, "Piotr's Image and Video Matlab Toolbox (PMT)," http://vision.ucsd.edu/ pdollar/toolbox/doc/index.html.