OBJECT TRACKING BY BIDIRECTIONAL LEARNING WITH FEATURE SELECTION

Heng Wang, Xinwen Hou, Cheng-Lin Liu

National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
95 Zhongguancun East Road, Beijing 100190, P.R. China
{hwang,xwhou,liucl}@nlpr.ia.ac.cn

ABSTRACT

This paper proposes a new tracking algorithm which combines object and background information, via building object and background appearance models simultaneously by non-parametric kernel density estimation. The major contribution is a novel bidirectional learning framework for discrimination between the object and background. It has the following advantages: 1) it embeds background information, unlike most other methods that focus on the object only, 2) it provides a mechanism to detect occlusion and distraction, which are two main causes of tracking failure, 3) it performs feature selection, making the tracker more robust to outliers. By this learning framework, we are able to embed discriminative information into the generative appearance model. Experimental results demonstrate that the tracker is able to model drastic appearance changes and robust to occlusion and distraction.

Index Terms— Tracking, appearance model, bidirectional learning, occlusion and distraction handling

1. INTRODUCTION

Object tracking has been widely studied during the last two decades. Generally, tracking can be formulated as how to find the object position given the object model and how to maintain the object model while the appearance of the object keeps changing. Recently, several new methods [1, 2, 3] model the object appearance in an online manner, unlike former works [4, 5] building the object model in advance. Jepson *et al.* [1] use density estimation to model the object as a mixture of probability distributions. The WSL tracker proposed in [1] involves three components. Unlike the WSL tracker fixes the number of components, Han *et al.* [2] build a Gaussian mixture model with adaptive number of components. Besides density estimation, Ross *et al.* [3] use a subspace to model the object. The object appearance is represented as the bases of the subspace, which are updated during the tracking process. The ground-breaking work [6] shows the value of background information, and introduces discriminative methods to the tracking literature. Avidan [7] uses boosting to train a classifier which is able to distinguish object pixels from background pixels. Here the object appearance model is implicitly maintained in the classifier.

The rest of the paper is organized as follows. In section 2, we introduce several related methods and show the differences and advantages of our approach. In section 3, we detail the flow of our algorithm. Experimental results are given in section 4. The paper is concluded in section 5.

2. RELATED WORKS

Appearance models in [1, 2] are both based on parametric methods, which can only model the density distribution of limited structure. In order to enhance the flexibility of the model, our method is based on non-parametric density estimation. Another difference is that former methods [1, 2] do density estimation for each pixel. This is suitable for tracking rigid objects, since it can embed spatial information in the appearance model. But for objects that undergo drastic shape distortion, it brings in unnecessary spatial constraint. So instead of modeling the distribution of each pixel, we model the distribution of the whole object area. Meanwhile, former density estimation methods [1, 2] only focus on the object, ignoring the background. In order to embed background information, our method models the appearance for both object and background.

Since the appearances of object and background keeps changing, we have to update the models while tracking proceeds. However, undesired updates may cause the model to drift away from the true density distribution. Further more, if it is left unresolved, the error will accumulate in the tracker, and finally cause tracking failure. In order to ease this problem, we design a bidirectional learning framework for model updating. We only select features that can help to discriminate the object from the background and discard features that may come from occlusion or distraction. By this way, we incorporate discriminative information into the generative density estimation method.

3. TRACKING ALGORITHM

In this section, we first briefly introduce density estimation, then we detail the flow of our tracking algorithm.

3.1. Density Estimation

For the object and background models, we use non-parametric density estimation, which is used in [8] for background subtraction. But they maintain a density distribution for each pixel, and set equal weights to all the features in the model.

For feature extraction, we divide the object and its nearby background area into equally sized overlapping blocks. For each block, we compute averaged R, G, B values as a feature vector. So when a new frame arrives, we have a batch of features. Let $x_1, x_2, \ldots, x_N \in \mathbb{R}^M$ be all the features accumulated up to now, and $w_i \in \mathbb{R}^1$ the corresponding weight, where M is the feature dimension. Given the feature set, the probability of a feature vector x_t at time t is as follows:

$$P(x_t) = \sum_{i=1}^{N} w_i K(x_t - x_i), \tag{1}$$

where K is a kernel function, and $\sum_{i=1}^{N} w_i = 1$, making all the features a distribution.

Most widely used kernels are the window function, the Epanechnikov kernel [9], the Gaussian kernel, etc. We choose the Gaussian kernel $N(0, \Sigma)$ as it can provide more accurate density estimation, where Σ represents the kernel function bandwidth. In order to decrease the computation burden, Σ is assumed to be a diagonal matrix. So the probability of observing x_t can be further formulated as:

$$P(x_t) = \sum_{i=1}^{N} w_i \prod_{j=1}^{M} \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{1}{2}\frac{(x_{tj}-x_{ij})^2}{\sigma_j^2}}, \tag{2}$$

where σ_j^2 are the entries of matrix Σ.

We build probability density models for the object and the background in the same way by extracting features from the object and the background area. So for each new feature x_t, we have $P^O(x_t)$, $P^B(x_t)$, which indicate the probability that the feature x_t comes from the object or the background. If there is no special instruction, we use superscripts O and B to denote the object and the background.

3.2. Flow of the tracking Algorithm

The framework of the tracking system is shown in Fig 1, and we summarize our tracking algorithm in Table 1. At the first video frame $\mathbb{I}_1$, the tracker is initialized by hand or some reliable detection results. Given the object position $\mathbb{L}_1$ in $\mathbb{I}_1$, the object and the background models are fed with features extracted from the object area and the nearby background. Object features $\{x_i^O\}_{i=1}^{N^O}$ and background features $\{x_i^B\}_{i=1}^{N^B}$ are set equal weights $\frac{1}{N^O}$ and $\frac{1}{N^B}$, where N^O and N^B are the numbers of features from object and background.

For each new frame $\mathbb{I}_t$, we extract features from previous object location $\mathbb{L}_{t-1}$, and compute $P^O(x_i)$, $P^B(x_i)$ for each feature x_i according to equation (2). Then $P^O(x_i)$ and $P^B(x_i)$ form the object confidence map C_t^O and the background confidence map C_t^B, respectively. In order to balance the weight of object model and background model, C_t^O and C_t^B are normalized to the same range $[0, 1]$. Finally, we get

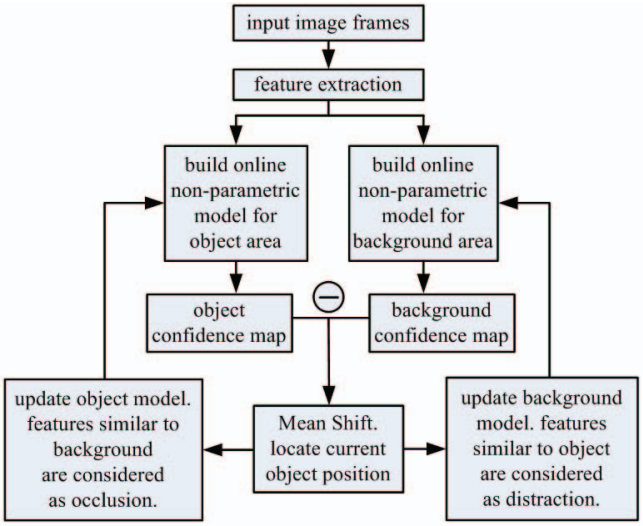

Fig. 1. The framework of the tracking system

the combined confidence map $C_t^{O\prime} - C_t^{B\prime}$, where $C_t^{O\prime}$ and $C_t^{B\prime}$ are the normalized object and background confidence maps. In the combined confidence map, if a feature has a positive value, it probably comes from the object, and its value indicates the likelihood that it is from the object.

We then locate current object position $\mathbb{L}_t$ by seeking the maximal value in the combined confidence map using Mean Shift [9]. Given the object position $\mathbb{L}_t$, we can label the features as object or background. However, if we directly add all these features to the object and background models, the tracker may suffer from model drift, as we can not guarantee 100 percent accuracy of the tracking results. In order to make the tracker more robust to drift, we propose the bidirectional learning rule, which is able to detect occlusion and distraction, and selects discriminative features for model update.

The bidirectional learning rule is formulated in table 2 and 3, where $P^O(x_i)'$ and $P^B(x_i)'$ are the normalized probability values. Occlusion and distraction are the two main reasons causing tracking failure. The former can disturb the tracker by lowering the probability values in the object area. The latter can absorb the tracker to a feigned object, which has a similar appearance as the true object. However, the bidirectional learning rule can help us to detect these two situations. For an object feature, if $P^B(x_i)'$ is much higher than $P^O(x_i)'$, this shows the feature may probably come from the background and occlude the object we want to track. For a background feature, if $P^O(x_i)'$ is much higher than $P^B(x_i)'$, this shows the feature has a high similarity with the object and could be a distractor for the tracker. We simply delete these kinds of features, as they may be occlusion or distraction.

For feature selection, we need to judge the discriminative power of a feature. We give high weights to object features with $P^O(x_i)'$ bigger than $P^B(x_i)'$, and background features with $P^B(x_i)'$ bigger than $P^O(x_i)'$. This is just a guideline which can be quantified to more specific formulas:

$$w_i^O = P^O(x_i)' \times (1 - P^B(x_i)'), \tag{3}$$

Algorithm Tracking Using Density Estimation
Input: Video frames $\mathbb{I}_t, t = 1, 2, \ldots, T$; Object location $\mathbb{L}_1$ in the first frame.
Output: Object location $\mathbb{L}_t, t = 2, \ldots, T$.
Initialization: For the first frame $\mathbb{I}_1$, extract object features $\{x_i^O\}_{i=1}^{N^O}$ and background features $\{x_i^B\}_{i=1}^{N^B}$. Add these features to the object or the background model with weights $\frac{1}{N^O}$ or $\frac{1}{N^B}$.

For each new frame $\mathbb{I}_t$, do:

1. Extract features from previous location $\mathbb{L}_{t-1}$. For each feature x_i, calculate $P^O(x_i)$, $P^B(x_i)$, and create the confidence map C_t^O for object, C_t^B for background.
2. Normalize C_t^O, C_t^B to the same range. Get $C_t^{O\prime}$, $C_t^{B\prime}$.
3. Run Mean Shift on the combined confidence map $C_t^{O\prime} - C_t^{B\prime}$. Obtain current object location $\mathbb{L}_t$.
4. Label the features as object or background according to object location $\mathbb{L}_t$.
5. Do feature selection using the bidirectional learning rule. Add new features to the object and the background model.
6. Update feature weights in the models. Delete features whose weights lower than a threshold, and normalize the weights of all the features to a distribution.

Table 1. The tracking Algorithm

$$w_i^B = P^B(x_i)' \times (1 - P^O(x_i)'), \tag{4}$$

where w_i^O and w_i^B are the object and background feature weights, and the value 1 is the upper bound, when we normalize the confidence maps.

After adding new features to the models, we update the weights of previous features as follows:

$$w_i^{O\prime} = \beta^O w_i^O, w_i^{B\prime} = \beta^B w_i^B, \tag{5}$$

where $\beta^O$and β^B are the learning rates of the object and background model. This can help the models to gradually forget previous features, and make a room for recent features. Finally, we delete features whose weights are lower than a threshold.

4. EXPERIMENTAL RESULTS

For all the experiments, we set the learning rates $\beta^O = 0.975$, $\beta^B = 0.85$, as generally the background changes more frequently than the object. We choose the block size for feature extraction to be 4×4 with $1/2$ overlapping. For the kernel bandwidth parameters, we simply follow the empirical formula in [10]: $\sigma = \theta / \log N$, where N is the feature number and θ the variance of the feature set. The width of the background area is $(h + w)/2$, with h and w the height and width of the object area. The algorithm is executed on a 1.86 GHz PC with 2G memory, and runs several frames per second.

$P^B(x_i)'$ \ $P^O(x_i)'$	low value	high value
low value	bad feature low weight	good feature high weight
high value	occlusion delete	bad feature low weight

Table 2. Bidirectional learning rule for object features

$P^B(x_i)'$ \ $P^O(x_i)'$	low value	high value
low value	bad feature low weight	distraction delete
high value	good feature high weight	bad feature low weight

Table 3. Bidirectional learning rule for background features

We do not use any motion assumption about the object, and the camera is not assumed to be fixed. The tracker is initialized in the first frame by hand, then it is working on its own. For qualitative comparison, we ran the Mean Shift tracker [9] using the same RGB feature as our method. In all the experiments, the object rectangle of our tracker is shown in red color, and the background rectangle in blue color, while the result of the Mean Shift tracker is in green color.

The first sequence is an indoor scene at the check-in of an airport from PETS2007[1].Our object is the head of a woman with a white dress, shown in Fig 2. The object undergoes large pose variations. Our tracker is able to follow it all along, while the Mean Shift tracker can not locate the object accurately in frame 828 and 878. The object is occluded by a man in frame 930, and both trackers are able to handle this occlusion. However, when it passing by another woman in frame1703, the Mean Shift tracker is absorbed by her face, and loses the true target. Our tracker follows the object through the distraction by the woman and the occlusion by the man in frame 1712. In frame 1974 the object is completely occluded by the man. This finally causes tracking failure.

The next sequence is from the same scene, but under a different view. We track the head of a woman crossing the crowd, depicted in Fig 3. Please note that the color of the woman's hair is similar to the color of the floor. When the object passes by another woman with the same hair color in frame 81, the Mean Shift tracker is completely absorbed by this feigned target. Our tracker survives in this challenge. The object is then occluded by a man in frame 92. Our tracker is able to follow the object through the whole scene.

The last sequence named "ThreePastShop1cor" is from the CAVIAR database[2] shown in Fig 4. We track the head of a man wandering in the corridor. The main challenge is large pose variances. The object turns left then right and meets

[1] ftp://ftp.pets.rdg.ac.uk
[2] http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

Fig. 2. Tracking a woman in complex background (Cropped for better view)

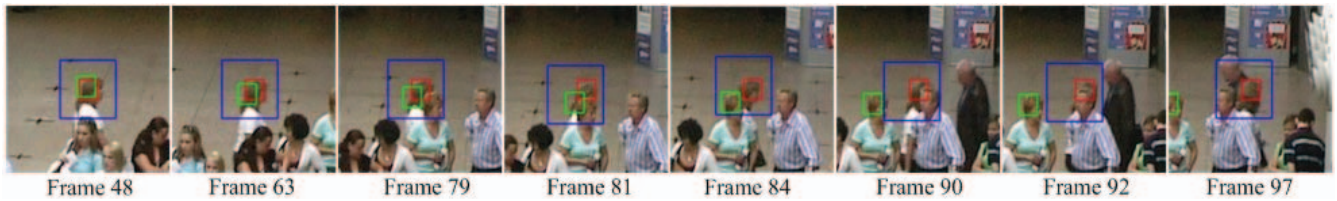

Fig. 3. Tracking a woman crossing the crowd (Cropped for better view)

Fig. 4. Tracking a man wandering in the corridor (Cropped for better view)

with another person. Both trackers are able to handle the pose variations, but the Mean Shift tracker drifts away to the pillar which has a similar color as the object in frame 263 and 273.

5. CONCLUSION

We propose a novel bidirectional learning framework which combines the object and background information together. The framework is able to detect occlusion and distraction, making the tracker robust to these outliers. Moreover, we incorporate feature selection in the tracker, which embeds discriminative information in the generative density estimation model. Online updating of the model enables our tracker to learn object appearance variations, and meanwhile robust to occlusion and distraction. Experimental results show that our method outperforms the Mean Shift tracker.

6. ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant nos. 60825301 and 60723005.

7. REFERENCES

[1] A.Jepson, D.Fleet, and T.El-Maraghi, "Robust online appearance models for visual tracking," *PAMI*, vol. 25, no. 10, pp. 1296–1311, 2003.

[2] B.Han, D.Comaniciu, Y.Zhu, and L.Davis, "Sequential kernel density approximation and its application to real-time visual tracking," *PAMI*, vol. 30, no. 7, pp. 1186–1197, 2008.

[3] D. Ross, J. Lim, R-S Lin, and M-H Yang, "Incremental learning for robust visual tracking," *IJCV*, vol. 77, no. 1, pp. 125–141, 2008.

[4] M. Black and A. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *IJCV*, vol. 26, no. 1, pp. 63–84, 1998.

[5] M.Isard and A.Blake, "Condensation-conditional density propagation for visual tracking," *IJCV*, vol. 29, pp. 5–28, 1998.

[6] R. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *PAMI*, vol. 27, no. 10, pp. 1631–1643, 2005.

[7] S. Avidan, "Ensemble tracking," *PAMI*, vol. 29, no. 2, pp. 261–271, 2007.

[8] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *ECCV*, 2000, pp. 751–767.

[9] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *PAMI*, vol. 25, no. 5, pp. 564–577, 2003.

[10] N. Kwak and C-H Choi, "Input feature selection by mutual information based on parzen window," *PAMI*, vol. 24, no. 12, pp. 1667–1671, 2002.