Online Spatial-temporal Data Fusion for Robust Adaptive Tracking

Jixu Chen Qiang Ji
Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA
chenj4@rpi.edu qji@ecse.rpi.edu

Abstract

One problem with the adaptive tracking is that the data that are used to train the new target model often contain errors and these errors will affect the quality of the new target model. As time passes by, these errors will accumulate and eventually lead the tracker to drift away. In this paper, we propose a new method based on online data fusion to alleviate this tracking drift problem. Based on combining the spatial and temporal data through a Dynamic Bayesian Network, the proposed method can improve the quality of online data labeling, therefore minimizing the error associated with model updating and alleviating the tracking drift problem. Experiments show the proposed method significantly improves the performance of an existing adaptive tracking method.

1. Introduction

In order to handle the appearance variance in object tracking, the target template or target model must be adapted over time. In recent years, many adaptive tracking techniques have been developed to cope with target pose change, illumination change or background clutters. These adaptive techniques can be classified into two main categories: online target model updating [8, 1, 7] and online feature selection [2, 13, 11]. The former performs online updating of target template, often in an incremental manner in order to develop a target model that best fits current target appearance. The methods in the second category perform online feature selection so that new features can be selected to better separate the target from the background. Examples of work in this category include Collins and Liu [2], Yin and Collins[13], and Wang and Chen [11]. Methods in both categories require to online and often on the fly relabel the image data using a classifier previously trained, and the labeled image data are then used to generate the new target model or to select new features. Due to limited discrimination capability with the classifier, labeling errors often occur, i.e., pixels are often mislabelled. And these mislabelled data are often ignored and are used to determine the new target model or new features. The mislabelled data will therefore pollute the new target model, and hence the quality of the new target model or new features will suffer due to these labeling errors. Each time the template is updated, small errors are introduced in target model. Over time, these errors will accumulate and eventually lead the tracker to drift away.

To alleviate this problem, in this paper we propose a new technique that improves the data labeling quality based on systematic integration of different measurements over time using a Dynamic Bayesian Network (DBN). The DBN allows to model the relationships between different measurements of a pixel and its label, as well as modeling the temporal evolution of each pixel. In our experiment, both the structure and the parameter of the DBN model are learned from large amount of training data. Given the learned DBN model and the measurements of each pixel, the label of each pixel can be determined by integrating different measurements over time through a probabilistic inference.

2. Literature Review

In most adaptive tracking method, it is common to observe the tracker drift due to incorrect adaptation, e.g., the target appearance model is adapted to background data or vice verse. Many methods have been proposed to mitigate the tracking drift problem.

In general the drift mitigation methods can be classified into two groups based on how the prior knowledge is used. The first group updates appearance model by anchoring each frame to the first frame [7, 2] or to a set of cases [14]. The underlying assumption of these methods is that the target should remain similar to either the off-line learned case-base or the target appearance in the first image frame. However, this assumption will be violated if target undergoes significant appearance change.

The second group [12, 1, 4, 9] does not need to obtain off-line learned prior knowledge. Instead, they can online update target model by imposing more constraints or assumptions in the updating process. .

Yang et al. [12] proposes to impose three constraints

1-4244-1180-7/07/$25.00 ©2007 IEEE

to improve the incremental subspace learning : (1) adaptive smoothness constraints, (2) negative data constraints, and (3) bottom-up pair-wise data constraints. With these constrains, they claim that the model can be updated more robustly. However,this method need to correctly label the background data and the target data online. If the labeled data contains error, the updated model will also contain errors.

In [4, 9], they use statistical model to retain an robust target template. The underlying assumptions of their method is that the appearance of the target should be temporally "stable". This assumption is reasonable in most cases. However if this assumption does not hold, e.g. a stable long time partial occlusion happens, it can not label the target pixel correctly, and the tracker will drift.

Avidan [1] maintains an ensemble of weak classifiers to classify the target pixels from background pixels. The ensemble is updated by training a new weak classifier on the current frame and adding it to the ensemble. Because some old classifiers are retained in the ensemble, the tracker drift problem is alleviated. When updating the new classifier, they also noticed that the tracker is very sensitive to outlier data. So they perform a simple outlier rejection process by setting a threshold on the pixel's weight, and then label the too "difficult" pixels in the tracking rectangle as background. The problem with this method is that, only the simple outlier rejection can not handle the data mislabeling problem very well.

We believe that one of the fundamental reasons for drift in adaptive tracker is the incorrect labeling. Model will drift because the target and background samples are mislabeled. However, few of current methods attempt to address the labeling problem.

Our data fusion method is applied to Collins and Liu's feature selection tracker in [2]. In this tracker, They have to online label both target and background data correctly to update model or select feature. So the labeling problem is especially important. Different from the previous work, we use an off-line learned Dynamic Bayesian Network (DBN) to robustly label the training data in current frame, thus improve both the subsequent online feature selection and the online template updating process.

3. Collins' Adaptive Tracking Method

In [2], Collins and Liu use feature selection to improve the tracking performance. The purpose of feature selection is to select the feature space that best distinguishes between object and its neighboring background.

3.1. Feature Selection

The set of candidate features consists of linear combinations of R,G,B values : $F = \{w_1R + w_2G + w_3B | w_* \in [-2, -1, 0, 1, 2]\}$. For each feature, they take the following steps to measure the separability of object and background pixels.

Feature selection process:

step 1. Estimate the distributions of object and background pixels with respect to the feature. Here, they use a "center-surround" approach to sample pixels from the object and the background, i.e., use a target rectangle detected in the previous frame to select all the pixels in that rectangle as target pixels, while a larger ring of neighboring pixels surrounding that rectangle is chosen to represent the background. From the extracted target and background samples, we can form the feature f's discrete p.d.f $p_f(i)$ for object, and the p.d.f $q_f(i)$ for the background, where i is the feature value.

step 2. Computing the log likelihood ratio of these distributions. For each feature value i, the likelihood ratio is defined as:

$$L_f(i) = log\frac{p_f(i)}{q_f(i)} \tag{1}$$

By this likelihood transformation, feature values appearing more often on target are mapped to positive values, and values that appear more often on the background are mapped to negative values. Thus ideally, in the likelihood space, both object and background have unimodel distribution.

step 3. The variance ratio is applied to this log likelihood to evaluate separability. This can be thought of an extended variance ratio(EVR). Let the overall combined density of the object and background be $p_f^{tot}(i) = (p_f(i) + q_f(i))/2$. Then extended variance ratio is defined as:

$$EVR_f(L; p, q) = \frac{var(L_f; p_f^{tot})}{var(L_f; p_f) + var(L_f; q_f)} \tag{2}$$

Where $var(L; p)$ denotes the variance of log likelihood function L with respect to distribution p [6]

Finally, the potential tracking features are ranked based on this EVR measurement to determine how well each feature distinguishes object from background in current frame. After ranking the features, top 5 most discriminative features are selected for subsequent mean-shift search process.

Note that the whole feature selection process is based on the current p.d.f $p_f(i)$ and $q_f(i)$. If we can not correctly extract target and background samples in *step 1*, e.g. some background pixels are included in target rectangle, then the likelihood function will contain errors and the features can not be ranked correctly.

3.2. Mean-shift Search

The above feature selection mechanism is based on the current frame. But we can assume the distribution of object and background features in the next frame remain similar to the current frame. So, first we use the top 5 features' log likelihood function $L_f(i)$ (Eq. 1) to transform the next

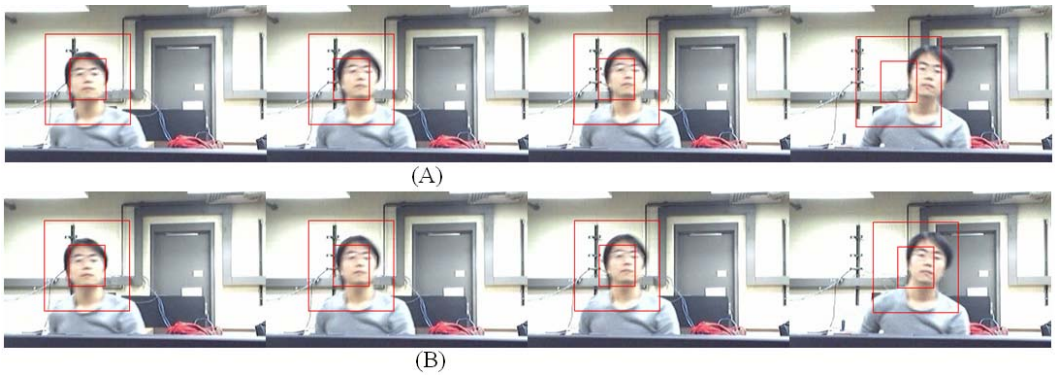

Figure 1. Tracking drift (The 0, 4, 8, 27 frame from 236 long sequence). (A) If we update current template in each frame, the tracker will drift gradually as background pixels start to "pollute" the target template. (B) Combining the current density with original density will alleviate the drift.

frame to 5 likelihood maps. Mean-shift search [3] is then performed on each of the five likelihood maps to detect the target. The final target position is based on combining the five target positions.

For the Mean-shift search, a target template need be constructed dynamically for each frame. This is accomplished based on the detected target in the previous frame. Specifically, a "distance weighted" histogram is used to construct the the target template. Let $\{\mathbf{x}_i\}_{i=1..N}$ be the locations of target pixels in previous frame, and the target center is $\mathbf{y_0}$, then the weighted histogram is constructed as:

$$p_u(\mathbf{y_0}) = C \sum_{i=1}^{N} k(\| \frac{\mathbf{y_0} - \mathbf{x}_i}{h} \|^2) \delta[b(\mathbf{x}_i) - u] \qquad (3)$$

where $u \in [1..m]$ is the index of feature values. Function $b : R^2 \rightarrow 1..m$ associates the pixel at location $(\mathbf{x}_i)$ with the index $b(\mathbf{x}_i)$ of feature value in that pixel. $k(x)$ is the Gaussian kernel profile. C is a normalization constant which impose the condition $\sum_{u=1}^{m} p_u(\mathbf{y_0}) = 1$.

In conventional adaptive tracker, this target template is updated using all pixels inside the target rectangle. However, because there may be some background pixels also included in target rectangle, this template updated process may introduce errors as shown in section 3.3.

3.3. Drift Problem

For each frame, new samples of target and background pixels are extracted for selecting features and for updating target template. However, adaptive updating both feature and target template in this manner raises the risk of drift. Figure 1 (A) shows an example of tracking drift. At first, only a few background pixels are included in target rectangle, but this error will cause a biased likelihood ratio in feature selection, and a biased target template in mean-shift search. So, in next frame more errors will be included. Finally, this error accumulation in each frame causes the tracker totally drift in Frame 27.

To handle the drift problem, in [2], they combine the current observed density with the original density in the first frame to give the current density function. In Figure 1 (B) we can see that this first frame compensated method can alleviate the drift problem. However, first this heuristic approach assumes that the object appearance will not change very much over the tracking sequence. Therefore, if target appearance changes significantly tracker will continue to drift. Second, this method do not try to prevent the background pixels to "pollute" the target template, it just down weights all the current data to alleviate drift.

4. The Proposed Method

Conventional adaptive tracker usually uses a target rectangle (or ellipse) to label the target pixels, i.e., label all the pixels in that rectangle as target and pixels outside that rectangle as background. As we can see in section 3.3, this kind of labeling will easily introduce errors, and finally cause the tracker drift.

We believe the tracker drift builds up as mislabeled "bad data" begun to pollute the model. So, the key issue addressed in this paper is robust labeling, i.e. label the object and background pixels robustly by integrating different measurements. Based on this robust label, we can train a more accurate appearance model and prevent the tracker drift.

Here we propose a method based on two improvements to alleviate tracking drift. The flowchart of our proposed algorithm is showed in Figure 2. In the first frame, given manually labeled training data, we first identify the top five features using the EVR measure. Target templates are then built for each of the five features. To construct the target template for each feature, instead of labeling every pixel inside the rectangle as target, only the pixels with a log likeli-

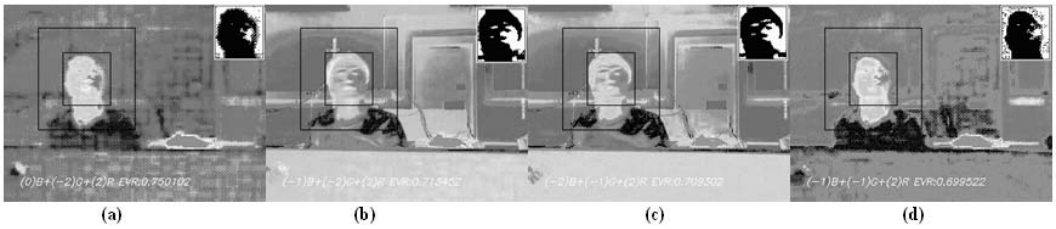

Figure 3. The top 4 likelihood maps of one frame and the selected target region for each map. (The selected target pixels in target rectangle are shown as dark region in upper right).

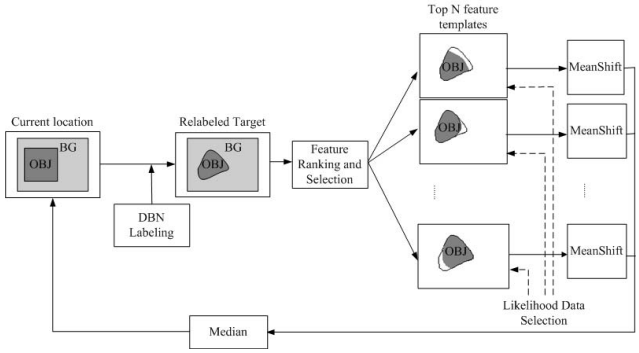

Figure 2. The flow diagram of DBN tracker

hood larger than zero are selected as the target pixels. Then, given the target template for each feature, mean-shift is used to identify the target location on the likelihood map of each feature. The final target location is taken as the median of the five target positions.

Second, for next frame, the image data need to be relabeled so that feature selection can be carried out again in the next frame. Instead of selecting the pixels within the target rectangle as target pixels and pixels outside the rectangle as background pixels, we introduce a data fusion method that combines measurements from different sources to improve the labeling process. Specifically, for each pixel, we have the following measurements: its intensity, its optical flow, its labeling in the previous frame, and its classification result by current top five features.(We will discuss these measurements in details in section 4.2.1.) They are systematically combined using a DBN and the probability of the pixel of being target after integrating these measurements is used to determine the final label of the pixel. The relabeled data are then used to train log likelihood and select the new features for the next frame. In the subsection to follow, we will discuss in more details of the proposed method.

4.1. Likelihood Data Selection

We first introduce a simple data selection in Collins tracker. In each of the top 5 likelihood maps, instead of selecting all pixels inside the target rectangle as target pixels, we only select the most "confident" target region to train the target template, which is used in subsequent mean-shift search.

The likelihood ratio $L_f(i)$ we introduced in section 3.1 gives us the cue to find the confident target. The feature values for target pixels tend to have positive likelihood ratio, and a large likelihood ratio means more confidence in target. On the other hand, the negative feature values tend to indicate the background pixels. Thus, we can select the only $L_f(i) > 0$ pixels in each of top 5 likelihood maps as target. Figure 3 shows the top 4 likelihood maps of first frame and the selected target region. We can see that in likelihood map Figure 3 (a) and 3 (d), the face region is most confident, but the hair part cannot be clearly separated from background, so we only select the face region to compute the target template. On the other hand, the left side of the face and the hair part are "confident" in Figure 3 (b) and 3 (c), so they are selected to compute target template in these two likelihood maps. Through this selection, we only include more "confident" pixels for constructing the target template, therefore minimizing the selection of the wrong target pixels.

4.2. DBN Labeling

4.2.1 DBN Model

As shown in section 4.1, likelihood data selection only select data in the template training step (as Figure 2), but do not select data in the feature ranking step. However, our experiment (section 5.2) shows that, if in feature ranking step we still select all the pixels in target rectangle to build the p.d.f of object $p_f(i)$ in Eq. 1, background pixels can also "pollute" the target rectangle seriously, then we may obtain wrong likelihood, and hence select wrong data in subsequent likelihood based data selection. So before feature selection, we first use a Dynamic Bayesian Network (DBN) to model different feature measurements for each pixel and to systematically combine the measurements to infer the true label for each pixel.

A DBN is dynamic version of the Bayesian Networks (BN). BN is a directed acyclic graph, with nodes representing the random variables and links representing the causal relationships among the nodes. DBN is constructed from

BN by cascading two BNs at two consecutive time slices. Like BN, DBN is parameterized by the conditional probability table for each node (CPT). For modeling the pixel labeling using DBN, we need first introduce the variables. There are five variables: L_j^t, L_j^{t-1}, Z_j^t, I_j^t, and OF_j^t. L_j^t and L_j^{t-1} represent the label of pixel j at time t and $t-1$. It has two values: target or background represented by 1 and 0 respectively. Because the label of each pixel changes smoothly, the labels L_j^t and L_j^{t-1} are correlated with each other. I_j^t is the intensity of pixel j at time t, Z_j^t is the labeling of pixel j at time t based on the top five features in time $t-1$, and OF_j^t is the optical flow of pixel j at time t. The value of each measurement node is determined as follows. For the Z_j^t node, its value is determined based on combining the classification results of the top five features through a majority voting scheme. Specifically, using the likelihood function of each feature to decide if a pixel should be classified as target or background. For example, if the log likelihood ratio is larger than 0, then that pixel is labeled as target or background otherwise. The final Z_j^t label for each pixel is then determined based on combining the classification results by top five features through a majority vote scheme as follows:

$$Z_j^t = \begin{cases} 1 & \text{more than two features classify it as target} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Because the top five features can best distinguish the background and target in last frame, the label Z_j^t is a good feature to distinguish the background and the target pixels in current frame. I_j^t is just the intensity of pixel j at time t. It can provide some additional information to classify target and background pixels. For OF_j^t, we employ the method [5] to compute the optical flow for each pixel in the image. OF_j^t is the magnitude of the optical flow vector for pixel j at time t. Since the background motion and the foreground motion are often very different, we believe optical flow measurement can provide additional information to help distinguish target from the background, especially when target and background appear similar.

Given the five variables, their relationships can be modeled by different DBNs. Figure 4 shows four possible DBN models. The directed links between the nodes represent the causal relations.Each model represents the relationships among the variables differently. For example, the model in Figure 4 (a) assumes the three measurements are conditional independent with each other given the label L. In next section, we will introduce a learning method to identify the best DBN model for the image data we use.

4.2.2 Learning DBN Model Structure

In order to explore the relationships among the five variables, it's necessary to use a large amount of training data

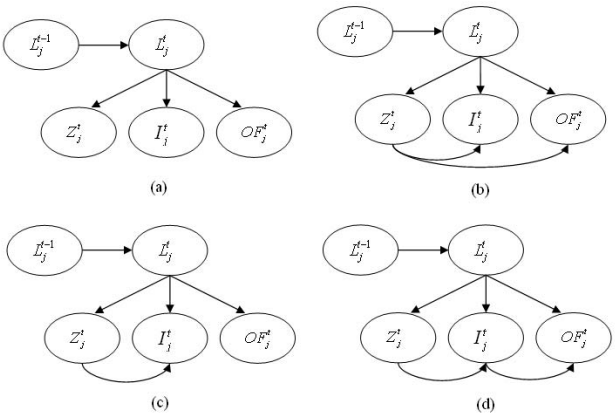

Figure 4. Some possible DBN structures can be used to infer the true label of current frame L_j^t. By DBN structure learning, we finally select the structure in (b) in our experiment

to identify the"correct" model for the data, through a structure learning algorithm.

The structure learning algorithm first defines a score that describes the fitness of each possible structure B_s to training data. Suppose we have a training data set $D = \{C_1, ..., C_n\}$, then a log likelihood score can be defined using the Bayesian information criterion (BIC) [10] :

$$\log p(D|B_s) \approx \log p(D|\hat{\theta}_{B_s}, B_s) - \frac{d}{2}\log(n) \tag{5}$$

where θ_{B_s} are the network parameters, $\hat{\theta}_{B_s}$ is the maximum likelihood estimate of θ_{B_s}, d is the number of logically independent parameters in B_s, and n is the size of D.

Then we can search for the optimal network structure with the highest BIC score. Given 5 nodes, there are totally 29,281 possible BN structures. However, in our case, we assume all the candidate structures are based on the Naive DBN as Figure 4 (a). So there are total of 25 different candidate structures, four of which are shown in Figure 4. From Equation 5, we can see that the first term of BIC score tends to select B_s that best fit the training data, and the second term tends to select B_s with simple structure. So, finally we can select the efficient and simple DBN structure. In our experiment, we collect the training data from a 100-frame manually labeled face tracking sequence, and the selected structure (the one with the highest BIC score) is shown in Figure 4 (b). Compared with the naive DBN structure in Figure 4 (a), there are two directed links from Z_j^t to I_j^t and from Z_j^t to OF_j^t are added to the naive structure. It means that the optical flow and intensity of each pixel are also related to its Z label.

4.2.3 DBN Inference

Given the DBN model, our purpose is to infer the current true label L_j^t given previous label and current measurements. Through an off-line training method, we can obtain the CPT for the DBN model shown in Figure 4 (b). Specifically, the conditional probability $P(Z_j^t|L_j^t)$, $P(I_j^t|Z_j^t, L_j^t)$ and $P(OF_j^t|Z_j^t, L_j^t)$ of this DBN are trained from the same 100-frames face tracking sequence in section 4.2.2. The translation probability $P(L_j^t|L_j^{t-1})$ is set manually as $P(L_j^t = 1|L_j^{t-1} = 1) = 0.6$ and $P(L_j^t = 1|L_j^{t-1} = 0) = 0.4$.

Given the parameterized DBN model, we can perform a probabilistic inference to obtain the probability of each pixel's label given its measurements and its label at previous time, i.e., we want to compute $P(L_j^t|L_j^{t-1}, Z_j^t, I_j^t, OF_j^t)$. From the DBN structure shown in Fig. 4 (b), this probability can be factorized as follows

$$P(L_j^t|L_j^{t-1}, Z_j^t, I_j^t, OF_j^t) = \alpha P(L_j^{t-1})P(L_j^t|L_j^{t-1})P(Z_j^t|L_j^t)P(I_j^t|Z_j^t, L_j^t)P(OF_j^t|Z_j^t, L_j^t) \quad (6)$$

where $P(L_j^{t-1})$ is the probability of the pixel j at previous time, which we already obtained at time $t-1$, $P(L_j^t|L_j^{t-1})$ measures the probability that the pixel label j makes a transition from time $t-1$ to t. It can be used to impose the temporal smoothness since the probability of a pixel changes label is small for a real time tracking. Probabilities $P(Z_j^t|L_j^t)$, $P(I_j^t|Z_j^t, L_j^t)$, and $P(OF_j^t|Z_j^t, L_j^t)$ are the marginal likelihood of the pixel label given each measurement. They are acquired through an offline training as mentioned before.

Based on the updated probability for $P(L_j^t|L_j^{t-1}, Z_j^t, I_j^t, OF_j^t)$, we then declare the pixel label for pixel j at time t as target if the probability is larger than a predetermined threshold.

4.3. Tracking Algorithm

In summary, the steps of the proposed method can be summarized as follows

Initialize

1) Manually select the first target rectangle r_1 in the first frame. Label pixels in r_1 as Z=1 and those outside it as Z=0.

Data selection

2) Use DBN to infer the probability $P(L = 1)$ for each pixel. Label the pixels with $P(L = 1) > \Theta$ as object (L=1) and pixels with $P(L = 1) <= \Theta$ as background (L=0). Then we have each features p.d.f on object and background. Further, we can compute each features likelihood Ratio $L_f(i)$ (Eq. 1) and EVR (Eq. 2).

3) Ranking and select the Top N features.

4) For each of the top N feature map, select the pixels whose likelihood>0 and L=1. Build the target template using the selected pixels.

Tracking

5) For next frame I_{i+1}, run mean-shift to find the nearest local mode in top N feature maps.

6) Merge N locations to give the new tracked rectangle r_{i+1}.

7) Go back to 2)

5. Experiments

In this section we present two tracking examples that illustrate the benefits of our data fusion method. In each experiment, the initial target is manually labeled in the first frame. For each of subsequent frames, our data labeling algorithm is performed to improve the feature selection and template updating process. In order to measure the tracking performance, the object center at each frame was manually identified, which serves as the ground truth.

5.1. Drift Mitigation

The first experiment uses a 236 frames sequence of a moving face. As shown in Figure 5 (A), although the Collins' tracker combined the original training data with the current observed data, it cannot handle the drift when the object appearance undergoes significant change, i.e. frame 112 and 119. In Figure 5 (B), it is obvious that our proposed DBN tracker can mitigate the drift in each frame by robustly label target and background pixels. In frame 112 and frame 119, although only a part of the face is observed, the DBN tracker correctly label this part as target and use these labeled data to update target model. The tracking error for every frame is shown in Figure 6. The horizontal axis is the frame number and the vertical axis is the pixel error for each frame compared with manually labeled target location. The solid line is the result of Collins tracker introduced in section 3. The dashed line is the result of our proposed DBN tracker. We can see that the Collins tracker begins to drift in frame 112, when the target appearance changes dramatically from the first frame.

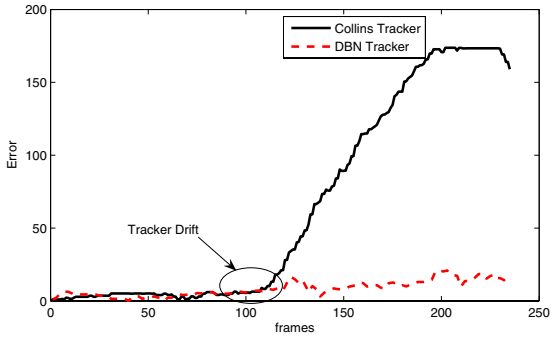

Figure 6. Comparison of the tracking performance when target undergoes a significant appearance change.

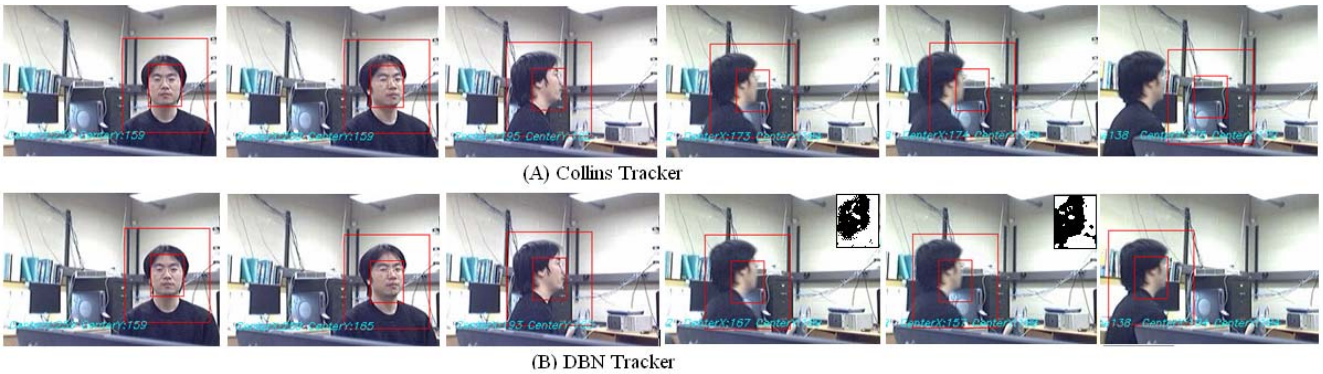

Figure 5. Comparison of the tracking results in face sequence 1. (A) The result of Collins tracker. (B) The result of proposed DBN tracker. The images are frame 1, 33, 92, 112, 119 and 138 from left to right. Especially for frame 112 and 119, the DBN labeled pixels in the target rectangle are shown in the upper-right of the image.

5.2. Performance under Partial Occlusion

Besides alleviating the drift problem as illustrated above, our DBN tracker can also handle partial occlusion during tracking well. In order to demonstrate that, a 160 frames long face video sequence with occlusion is recorded, as shown in Figure 7.

Two other trackers are applied in this sequence as well. The first one is the Collins tracker as we used before. The second one uses the same likelihood data selection method, but it does not apply DBN data selection before feature selection, i.e., this likelihood tracker still use all the pixels in rectangle to train likelihood and select feature.

For Collins tracker, as shown in Figure 7 (A), it drifts away gradually as more and more background pixels are included in target rectangle. For likelihood tracker (Figure 7 (B)), we can see that at first, the book is taken as background, and is easy to be separated from the target in top 1 feature map. But when the face is occluded by the book gradually, more and more pixels of the book are included in the target rectangle. When a large part of the target rectangle is occupied by the book, the tracker began to assume the book as target, and in the new selected top 1 feature's likelihood map, the book is turned to be more obvious than the face. Finally, the tracker is attracted to the book.

The performance of the proposed DBN tracker is shown in Figure 7 (C). We can see that in frame 71 and 79, although a large part of face is occluded, the tracker can also only label the face pixels as target, without being attracted to other objects.

The pixel errors for 160 frames long sequence are shown in Figure 8. It demonstrates that our DBN tracker works well under partial occlusion.

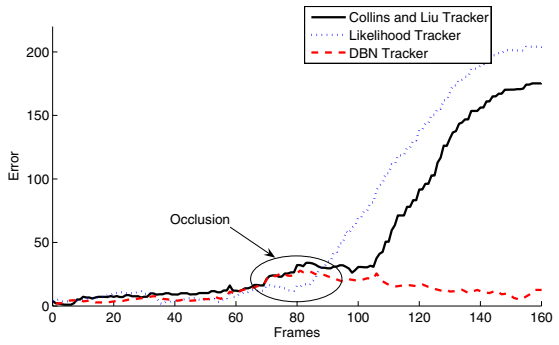

Figure 8. The tracking performance of partial occluded face sequence 2.

6. Conclusion

Compared with Collins method, the main innovations of the proposed work lie in three aspects: first the log likelihood is used to select the target pixels for constructing the target template. Second, a DBN model is introduced to integrate different pixel measurements to improve the labeling process. Third, we learned both the structure and the parameter of the DBN model, so the learned DBN can be both representative of the data and simple. These three improvements result in significant performance improvement as demonstrated by our experiments. While the proposed method is specifically for Collins method in this paper, we believe the basic idea can be extended to other adaptive tracking methods as well.

References

[1] S. Avidan. Ensemble tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:494–501, 2006. 1, 2

[2] R. Collins and Y. Liu. On-line selection of discriminative tracking features. *IEEE International Conference on Computer Vision*, 1:346, 2003. 1, 2, 3

[3] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. *IEEE Conference on*

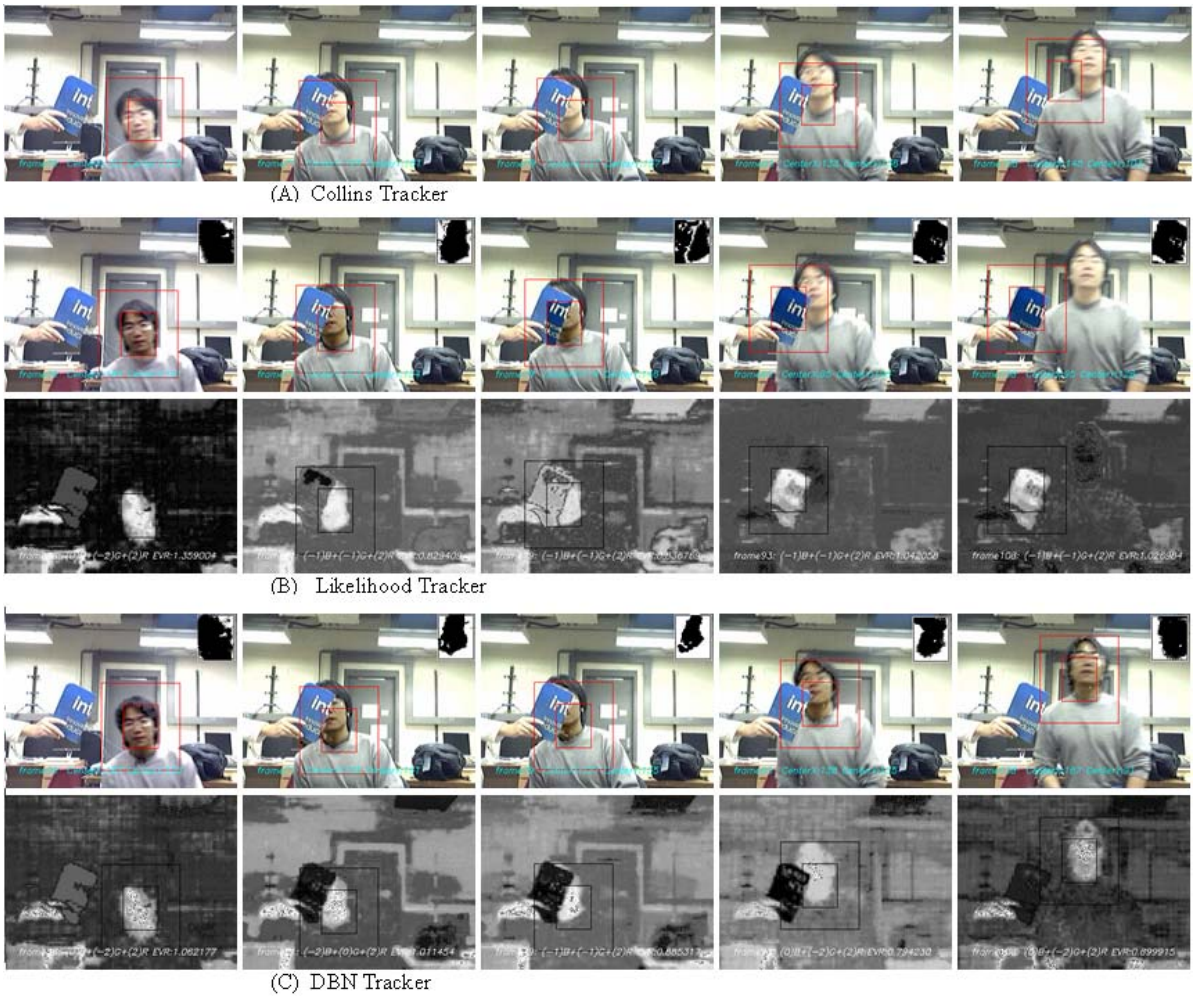

Figure 7. Comparison of the tracking result in partial occluded face sequence 2. (A) The result of Collins tracker. (B) The result of Likelihood tracker. The first row shows the tracked object (The selected pixels by the top 1 feature are shown in upper right rectangle). The second row shows corresponding likelihood image of the top 1 feature. (C) The result of the proposed DBN tracker. The first row shows the tracked object and the second row shows the top 1 feature's likelihood image after DBN labeling. The images are frame 31th, 71th, 79th, 93th and 108th from left to right.

Computer Vision and Pattern Recognition, 2000. 3

[4] A. D.Jepson, D. J.Fleet, and T. F.El-Maraghi. Robust online appearance model for visual tracking. *CVPR01*, 1:415–422. 1, 2

[5] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981. 5

[6] A. F. Karr. *Probability (Springer Texts in Statistics)*. Springer Verlag, 1996. 2

[7] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. In *Proceedings of the British Machine Vision Conference*, September 2003. 1

[8] H. Nguyen, Q. Ji, and A. Smeulders. Spatio-temporal context for robust multi-target tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29, 2007. 1

[9] H. T. Nguyen and A. W. Smeulders. Fast occluded object tracking by a robust appearance filter. *PAMI*, 26:1099–1104, 2004. 1, 2

[10] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978. 5

[11] J. Wang, X. Chen, and W. Gao. Online selecting discriminative tracking features using particle filter. *CVPR05*, 2:1037–1042. 1

[12] M. Yang and Y. Wu. Tracking non-stationary appearances and dynamic feature selection. *CVPR05*, 2:1059–1066. 1

[13] Z. Yin and R. Collins. Spatial divide and conquer with motion cues for tracking through clutter. *CVPR06*, 1:570–577. 1

[14] Z. Zhu, W. Liao, and Q. Ji. Robust visual tracking using case-based reasoning with confidence. *CVPR06*, 1:806–816. 1