Robust Visual Tracking via Pixel Classification and Integration

Cha Zhang and Yong Rui
Microsoft Research, One Microsoft Way, Redmond, WA 98052
{*chazhang,yongrui*}*@microsoft.com*

Abstract

We propose a novel framework for tracking non-rigid objects via pixel classification and integration (PCI). Given a new input frame, the tracker first performs object classification on each pixel and then finds the region that has the highest integral of scores. There are several key advantages of the proposed approach: it is computationally very efficient; it finds a global, instead of local (e.g., mean-shift), optimal solution within a search range; and it is inherently robust to different object scales with minimum extra computation.

Within this framework, a mixture of long-term and short-term appearance model is further introduced to perform PCI. As a result, the tracker is able to adapt to both slow and rapid appearance changes without drifting. Challenging video sequences are presented to illustrate how the proposed tracker handles large motion, dramatic shape changes, scale variations, illumination variations and partial occlusions.

1. Introduction

Real-time object tracking has been a critical component in many computer vision applications such as surveillance, distributed meetings, video compression, etc. There have been many existing algorithms for object tracking in literature. In particular, the mean-shift (MS) algorithm [1], including its variations such as [2], is one of the best for non-rigid object tracking tasks. As a gradient-based approach, MS is very efficient and easy to implement. It does a good job in tracking objects with partial occlusions, background clutters, large motions, etc. Despite many of its good properties, it has several limitations. First, like all gradient-based approaches, MS can only find a local optimal object position instead of a global one. This is a well known issue and was alleviated using multi-bandwidth mean shift procedure by Shen et al. in [12] . Second, MS does not perform well for scale variations. Not only it needs to perform at different scales, as pointed out in [5], the MS algorithm tends to shrink the tracking region when the object increases its size. Lastly, MS does not have an efficient appearance modeling mechanism, and is sensitive to illumination variations and severe partial occlusions.

In this paper, we first present a tracking framework via pixel classification and integration (PCI), which solves the first and second problems mentioned above (Section 2 and 3). The basic idea is to soft-classify each pixel of the current frame into foreground and background, then search for a region that contains the most foreground pixels. We demonstrate the framework in Section 3 with a simple per-pixel color classification based tracker. By using the concept of integral image [3], the proposed algorithm is more efficient than MS, and it guarantees to find the global optimal object location within the region of interest. Moreover, scale adaption can be performed accurately with minimum additional cost under the proposed framework.

We address the third problem by using appearance modeling for color histogram based trackers. A generative histogram model is introduced as a mixture of a long-term component and a short term component in Section 4. The mixing weights reflect how good each component can describe the tracked object in the current frame. By combining the appearance modeling and the color PCI tracker mentioned above, we are able to track some very challenging sequences, as shown in Section 5.

2 PCI Based Visual Tracking

We first present a tracking framework that performs classification on each pixel of the current frame and finds the tracked object by integrating all the classification results together. Consider a pixel in the current frame, denoted as $\mathbf{c}(\mathbf{x})$, where $\mathbf{c}$ is a vector describing the pixel's features such as color and/or texture patterns; $\mathbf{x}$ is the pixel location. Assume that a foreground/background classifier $\mathscr{C}$ is available. We perform classification on all the pixels, which yields:

$$s(\mathbf{x}) = \mathscr{C}\big(\mathbf{c}(\mathbf{x})\big). \tag{1}$$

where $s(\mathbf{x})$ is the output score of the classifier, normally a real value indicating how likely the pixel belongs to the foreground object. We then determine the foreground region in the current frame as:

$$\begin{aligned} R_f &= \arg\max_R J(R) \\ &= \arg\max_R q(R) \sum_{\mathbf{x}\in R} s(\mathbf{x}). \end{aligned} \tag{2}$$

where R_f is the new foreground region; $J(R)$ is the target function we want to optimize. $\sum_{\mathbf{x}\in R} s(\mathbf{x})$ is the integral of

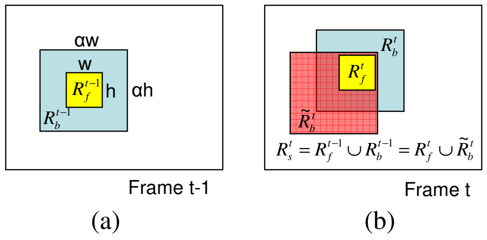

Figure 1. The regions defined for the color histogram classifier. (a) For frame $t-1$, R_f^{t-1} is the object region; R_b^{t-1} is the surrounding background region. (b) Frame t after tracking. R_s^t is the extended search region. R_f^t is the new tracked object region and R_b^t is the new background region; $\widetilde{R}_b^t$ is the *updating* background region used for updating the mixing weights in Section 4.

the score image within region R. We assume the object has a rectangular region of support, and its size is fixed (scale adaptation is discussed in Section 3.2). $q(R)$ is a probability that indicates how likely region R is the foreground region, which can take object temporal consistency or motion prediction into the tracking framework.

Once the output score surface $s(\mathbf{x})$ has been calculated, optimizing Equation 2 can be done through exhaustive search, thanks to the special form of Equation 2. We first compute the integral image [3] of the score surface in the searching area. For any rectangle region R, $\sum_{\mathbf{x}\in R} s(\mathbf{x})$ can be calculated with 3 additions, which is super-efficient. An important benefit of exhaustive search is that it guarantees a global optimal solution rather than a local one such as mean-shift.

3 A Color PCI Tracker

We next present a color classification based tracker using the framework described above. We use the pixel color as the feature $\mathbf{c}(\mathbf{x})$ of our classifier, i.e., $\mathbf{c}$ is a three dimensional vector representing the YUV color of the pixel at $\mathbf{x}$.

3.1 The Classifier and the Tracker

When classifying the foreground object from the background, it certainly helps if we can model the foreground and the background jointly, as the discrimination between them is one of the most important factors that can affect the tracking performance. This hypothesis was successfully used in [8] for the on-line selection of tracking features. In the proposed color PCI tracker, we build two color histograms – one models the foreground object and the other models the background objects. Let $u = 1, 2, \cdots, m$ be the bins of the histograms. To simplify notation, we use $u(\mathbf{x})$ to represent the color bin that pixel color $\mathbf{c}(\mathbf{x})$ falls in. Hence our classification feature is a one dimensional variable for each pixel location $\mathbf{x}$.

Once a foreground object is given in the first frame, we compute its histogram in that frame. Denote it as $p_f(u)$, where subscript f stands for the *foreground* object. In contrast, the background histogram is always constructed from the previous frame during the tracking process. Let the region of tracked object in frame $t-1$ be R_f^{t-1}. We first expand the object region by α in each direction, where α is a scalar whose typical value is 3. We then compute the color histogram of the surrounding *background* region R_b^{t-1} (Figure 1(a)), denoted as $p_b^{t-1}(u)$. The whole region $R_s^t = R_f^{t-1} \cup R_b^{t-1}$, defines the the *search* region of the tracker (Figure 1(b)). For frame t, The classification score of a pixel at $\mathbf{x}$ is computed as:

$$s^t(\mathbf{x}) = \log \frac{p_f(u(\mathbf{x}))}{p_b^{t-1}(u(\mathbf{x}))}, \mathbf{x} \in R_s^t, \tag{3}$$

which is the log likelihood ratio of the pixel belonging to the foreground and the background. According to Equation 2, the tracked object region in the next frame is computed as:

$$\begin{aligned} R_f^t &= \arg\max_R J^t(R) \\ &= \arg\max_{R\subset R_s^t} q^{t-1}(R) \sum_{\mathbf{x}\in R} s^t(\mathbf{x}), \end{aligned} \tag{4}$$

where $q^{t-1}(R)$ models the temporal relationship between previous object locations and the current one. Currently we model such a probability with a simple multivariate Epanechnikov kernel [4]:

$$q^{t-1}(R) = \max\left(0, 1 - d^2(R, R_f^{t-1})\right), \tag{5}$$

where $d(R, R_f^{t-1})$ is the normalized Euclidean distance between the centers of R and R_f^{t-1}. The intuition here is that we prefer locations that are near the foreground object's previous position. Let the tracked object have width w and height h, and the centers of R and R_f^{t-1} differ by $(\Delta x, \Delta y)$, we have:

$$d(R, R_f^{t-1}) = \sqrt{\left[\frac{2\Delta x}{(\alpha-1)w}\right]^2 + \left[\frac{2\Delta y}{(\alpha-1)h}\right]^2}. \tag{6}$$

For a typical value of $\alpha = 3$, the normalization factors simply become w and h. Note that more complex motion models can be applied for $q^{t-1}(R)$ when necessary.

The color PCI tracker described above resembles partly with the work in [11], where a similar color log likelihood image is adopted to update the sample weights in a particle filtering framework. In this work the likelihood image is used directly to locate the object. We find the temporal term $q^{t-1}(R)$ in Equation 4 is of great importance to enhance the performance of likelihood images in clutter. This is demonstrated by the successful tracking of the football sequence shown in Section 5.

3.2 Scale Adaptation

The scale of the target object often changes with time in real world, which makes it necessary to adapt the size of the tracked object region accordingly. Unfortunately, the integral of classification scores measured in Equation 4 is not scale invariant. The scale adaption problem is solved by introducing a

Difference of Gaussian (or normalized Laplacian of Gaussian) kernel [5], we approximate such a kernel with the following optimization problem:

$$\begin{aligned} R_f^t &= \arg\max_R J_{scale}^t(R) \\ &= \arg\max_{R\subset R_s^t} q^{t-1}(R)\frac{1}{w\cdot h}\Big[\sum_{\mathbf{x}\in R} s^t(\mathbf{x}) - \gamma \sum_{\mathbf{x}\in R_\beta} s^t(\mathbf{x})\Big], \end{aligned} \tag{7}$$

where $w \cdot h$ is the size of region R; R_β is a slightly expanded region of R. γ is a parameter one can tune to adjust the importance of foreground and background objects. We choose $\beta = 0.2$ and $\gamma = 0.6$ in our experiments, although γ's within $(0.4, 0.7)$ all provide satisfactory tracking results.

Comparing with Equation 4, for each candidate region R, Equation 7 introduces 4 additional additions and 1 additional multiplications when the same integral image based method is used, which is quite small. We optimize Equation 7 for 5 times with different scales $(0, \pm 5\%, \pm 10\%$ size variation) and pick the scale that gives the highest $J_{scale}^t(R)$. It's worth pointing out that the same integral image can be reused while trying different scales, which is much more efficient than running the whole tracker 5 times, as is necessary in the mean-shift algorithm [1].

4 Appearance Modeling

Another important aspect of a robust tracker is its ability to adapt to appearance changes due to illumination variations and partial occlusions. Work has been done to model color variations under different lighting conditions [6, 7], but they either target for a certain type of object (such as skin) or have high computational cost. In [13], Nummiaro et al. update the target model by taking the weighted average of the current and new histograms. Han and Davis [14] proposed a sequential density approximation based method using Gaussian mixture as the target model. In this section, we present a robust appearance modeling for the model histograms used in the above color PCI tracker. It can also be considered as an advanced classifier for the color PCI algorithm presented above. Note that the algorithm described here can also be applied to other color histogram based trackers, such as the mean-shift algorithm.

We formulate a generative histogram model as a mixture of two components, namely, a long-term component that changes slowly in time, and a short-term component that adapts rapidly to the most recent appearance of the object. Under normal situation, we expect the long-term component to take the lead in determining where the object goes. When there are lighting variations or partial occlusions, the short-term component should play an important role to quickly follow the appearance changes of the tracked object. Similar ideas were introduced in [9], where they have three components instead: stable, wandering and lost. Their stable component is similar to our long-term component, and their wandering component corresponds to our short-term component. We do not have a lost component. In [9], the lost component is used to model rapid feature variations that cannot be modeled by either the stable or the wandering component. In our approach, such rapid variation is rare because the object histogram is a region-based feature.

The foreground and background histogram can now be written as:

$$p_f(u) = m_{fl}p_{fl}(u) + m_{fs}p_{fs}(u) \tag{8}$$

$$p_b(u) = m_{bl}p_{bl}(u) + m_{bs}p_{bs}(u) \tag{9}$$

where $p_f(u)$ and $p_b(u)$ are the foreground and background histograms, respectively. $p_{fl}(u)$ and $p_{fs}(u)$ are also histograms, where $p_{fl}(u)$ stands for the long-term component of the foreground histogram and $p_{fs}(u)$ stands for the short-term component. m_{fl} and m_{fs} are the mixing weights, which satisfies $m_{fl} + m_{fs} = 1$. These weights indicates how important different components are to the current appearance. Similarly, $p_{bl}(u)$ and $p_{bs}(u)$ are the long-term and short-term components of the background histogram. Mixing weights $m_{bl} + m_{bs} = 1$.

At the first frame ($t = 0$) of the video sequence, we initialize $p_{fl}^0(u)$ and $p_{fs}^0(u)$ to be identical to the histogram obtained from the to-be-tracked object region R_f^0. The background histograms $p_{bl}(u)$ and $p_{bs}(u)$ are also identical to the histogram computed from R_b^0 (Figure 1). The mixing weights are initialized as $m_{fl} = 0.8$, $m_{fs} = 0.2$, $m_{bl} = 0.2$ and $m_{bs} = 0.8$, although they can be chosen arbitrarily as they will adapt to the sequence very quickly.

Now assume we have all the information at time $t-1$, and we want to track the object at time t. The first step is to compose $p_f^{t-1}(u)$ and $p_b^{t-1}(u)$ as in Equation 8 and 9. Then, we may use the color PCI tracker introduced in Section 3.1 to get the new foreground region R_f^t, as shown in Figure 1(b). Since we mentioned that the mixing weights indicate how good each component is in modeling the current appearance, we can use the Kullback Leibler distances [10] to measure how good a component explains the object appearance. For the foreground object, we compute a color histogram from R_f^t in Frame t, denoted as $\widetilde{p}_f^t(u)$[1]. Let:

$$d_l = \sum_u \widetilde{p}_f^t(u) \log \frac{\widetilde{p}_f^t(u)}{p_{fl}^{t-1}(u)} \tag{10}$$

$$d_s = \sum_u \widetilde{p}_f^t(u) \log \frac{\widetilde{p}_f^t(u)}{p_{fs}^{t-1}(u)} \tag{11}$$

where d_l and d_s are the Kullback Leibler distances between $\widetilde{p}_f^t(u)$ and the two component histograms. The new weights are:

$$\widetilde{m}_{fl}^t = \frac{e^{-d_l/\sigma}}{e^{-d_l/\sigma} + e^{-d_s}} \tag{12}$$

[1]To clarify the notations, all intermediate histograms or parameters used for weight updating will have the form $\widetilde{a}$; and all intermediate histograms or parameters used for component histogram updating will have the form $\hat{a}$.

$$\widetilde{m}_{fs}^t = \frac{e^{-d_s}}{e^{-d_l/\sigma} + e^{-d_s}} \quad (13)$$

$$m_{fl}^t = \eta\widetilde{m}_{fl}^t + (1-\eta)m_{fl}^{t-1} \quad (14)$$

$$m_{fs}^t = \eta\widetilde{m}_{fs}^t + (1-\eta)m_{fs}^{t-1} \quad (15)$$

where $\widetilde{m}_{fl}^t$ and $\widetilde{m}_{fs}^t$ are the relative goodness of each component (normalized). The actual mixing weights m_{fl}^t and m_{fs}^t are updated in a linear regression manner, with the default value of $\eta = 0.1$. Experiments show that the algorithm operates with a range of η. σ reflects the prior belief on the long-term component over the short-term component. The default value of σ is 5. When the long-term component and the short-term component have the same distance to $\widetilde{p}_f^t(u)$, we prefer the long-term model to explain the coherent observation in order to prevent the model from drifting. A similar arrangement was made in [9] for the same purpose.

The background mixing weights are computed in a similar way, except that we replace $\widetilde{p}_f^t(u)$ with $\widetilde{p}_b^t(u)$, where $\widetilde{p}_b^t(u)$ is a histogram collected from the region $\widetilde{R}_b^t$, as shown in Figure 1(b). Note that the mixing weights we calculate here serve the purpose of predicting the background histogram in the next frame, which could be counter-intuitive.

The final step is to update the component histograms. We compute a histogram $\hat{p}_f^t(u)$ from region R_f^t (in this case $\hat{p}_f^t(u)$ is identical to $\widetilde{p}_f^t(u)$), and update the foreground components as:

$$p_{fl}^t(u) = \rho_1\hat{p}_f^t(u) + (1-\rho_1)p_{fl}^{t-1}(u) \quad (16)$$

$$p_{fs}^t(u) = \rho_2\hat{p}_f^t(u) + (1-\rho_2)p_{fs}^{t-1}(u), \quad (17)$$

where ρ_1 and ρ_2 are two parameters controlling the update speed. Due to the characteristics of long-term and short-term component, we use $\rho_1 = 10^{-4}$ and $\rho_2 = 0.9$. Similarly, we compute a background histogram $\hat{p}_b^t(u)$ from region R_b^t (Figure 1(b)). The background components are updated as:

$$p_{bl}^t(u) = \rho_1\hat{p}_b^t(u) + (1-\rho_1)p_{bl}^{t-1}(u) \quad (18)$$

$$p_{bs}^t(u) = \rho_2\hat{p}_b^t(u) + (1-\rho_2)p_{bs}^{t-1}(u). \quad (19)$$

5 Experiments

The proposed tracker runs comfortably in real time on a Pentium 4 2.8 GHz computer (3-10% of CPU usage) without further optimization. In this section, we compare the PCI tracker with the mean-shift (MS) tracker and an extended MS version which does background modeling (MSBM) [1] on four different sequences:

1. *Football* sequence. It was used in the original mean-shift paper [1], hence we use it here for a sanity check. This sequence demonstrates that the color PCI tracker can handle large object and camera motions, motion blur, dramatic shape changes (caused by player's non-rigid motion), distractors (same team players) and partial occlusions.

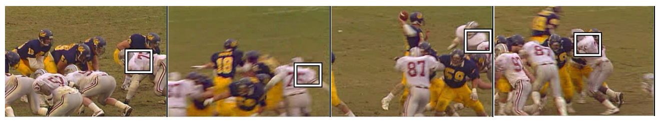

Figure 2. *Football* sequence, tracking player 75. The frames 1, 16, 30, 34 are shown. Note the sequence is at 5 fps and heavily compressed before tracking.

2. *Walking* sequence. It features large scale variations. The foreground size is between 15×50 and 66×246.
3. *Pedestrian* sequence. It is captured from a surveillance camera and the pedestrian is very small (7×16) in the view. The most difficulty part of the sequence is a heavy partial occlusion period when the pedestrian walked behind a bush.
4. *Office* sequence. This is the most challenging one. There exists background distractors (door), fast motions, partial or complete occlusions (mounting a book in front of the face, hiding behind a chair), slow and dramatic light variations (turn on/off lights), etc.

In all the sequences that follow, we adopt the same default values of the parameters in the tracker. The YUV color space is quantized into $16 \times 16 \times 16$ bins, and the search expansion factor $\alpha = 3$.

Figure 2 shows the tracking results on the *football* sequence. The sequence resolution is 351×240. In the first frame, player 75 is selected as the foreground object. Despite large motions, motion blur, dramatic shape changes, distractors (frame #30), and partial occlusions, the tracker successfully tracks through the sequence. Note the *football* sequence we use here is at 5 fps and heavily compressed, which makes the tracking task even harder.

Figure 3 shows the performance of the proposed tracker on the *walking* sequence. A person walks around a building and his appearance has huge scale variations. The sequence has 640×480 resolution and 15 fps. The result is compared with that of a MS tracker with scale adaption [1]. Both trackers try 5 different scales for each input frame. Note the MS tracker is much more expensive compared with PCI. The MS tracker does a reasonable job while the person walks away from the camera, but it failed when the person walks back toward the camera. Similar effects was observed in [5]. The proposed PCI algorithm tracks the person very well throughout the whole sequence.

Figure 4 shows the tracking performance on the *pedestrian* sequence using the proposed color PCI tracker with appearance modeling. The resolution of the video is 640×480, and the region of the tracked pedestrian is small. We compare our tracker with MS and MSBM. At frame #130, the pedestrian walks behind a bush, which introduces heavy partial occlusion. Both MS and MSBM lost the pedestrian at this moment. The regular MS algorithm loses the pedestrian hereafter, while the MSBM algorithm is able to re-lock the tracked subject after a

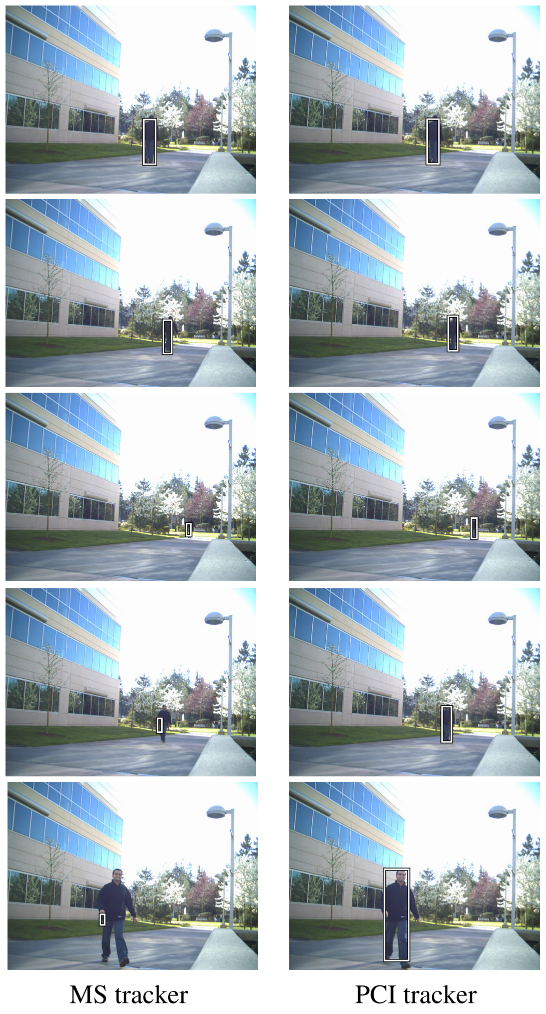

Figure 3. *Walking* sequence. The frames 2, 55, 175, 340 and 440 (top to bottom) are shown.

while. Our appearance modeling assisted tracker successfully tracks the pedestrian throughout the whole sequence.

To provide some insights on why the proposed tracker works, Figure 5 shows the mixing weights of the appearance model. The black bold solid curve is the weight for the foreground long-term component, and the red bold dashed curve is that for the foreground short-term component. Note these two curves sum up to 1 at any time instance. We draw them both for reader's convenience. Around frame #130, the short-term component's weight increases dramatically to compensate for the partial occlusion effect. This explains why the proposed tracker can still track the pedestrian despite the heavy occlusions.

In Figure 6, we show the result on the *office* sequence. It is captured at 24 fps and has resolution 320 $\times$ 240. Again we compare the color PCI tracker with appearance modeling to the regular MS algorithm and the MSBM algorithm. At frame #220, all three trackers still track the face after a complete occlusion by a book. At frame #320, the subject turns off the light. At that instance, all the three trackers are still tracking

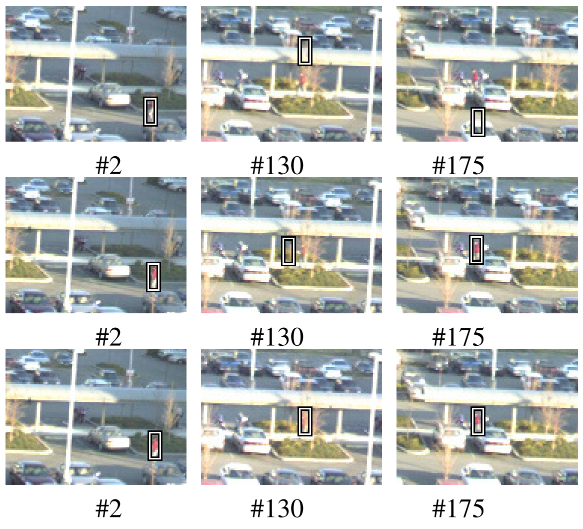

Figure 4. *Pedestrian* sequence. The frames 2, 130 and 175 are shown. The top row is the result of MS; the second row is the result of MSBM; the bottom row is the result of PCI with appearance modeling. Note the original video is at 640 $\times$ 480 resolution, and the images shown above are cropped to show the region of interest.

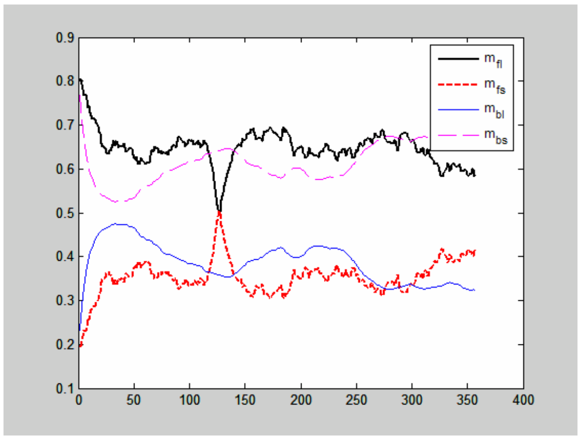

Figure 5. The mixing weights of the appearance model for the *pedestrian* sequence.

the head. PCI continues to track the head region under very low light condition (frame #360), while MS and MSBM fail because of the lack of appearance adaptation. The subject then turns the light on. After a while, both MS and MSBM re-lock the head region (frame #560) as the face moves close to the tracking region. Unfortunately, at frame #620, both MS and MSBM are distracted by the hand which has similar color distributions as the head. PCI works surprisingly well and is able to track the subject throughout the whole sequence.

The mixing weights of the appearance model are shown in Figure 7. The first thing one may notice is, compared with Figure 5, the average mixing weights of the object long-term component and object short-term component are about half-half. This shows that the foreground's color distribution is varying more than the Pedestrian sequence. The mixing weight

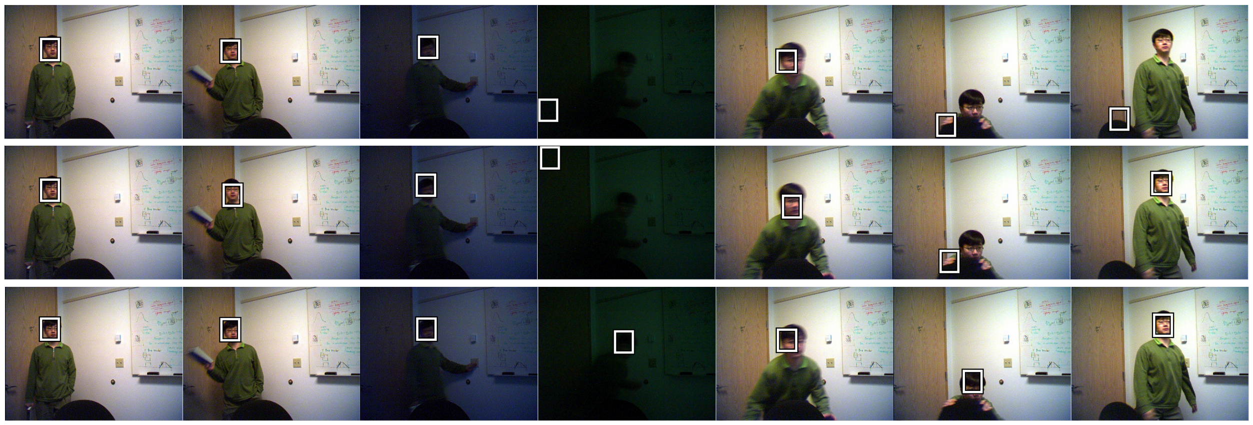

Figure 6. *Office* sequence. The frames 2, 220, 320, 360, 560, 620 and 650 are shown. The top row is the result of a MS tracker; the middle row is the result of a MSBM tracker; the bottom row is the PCI tracker with appearance modeling.

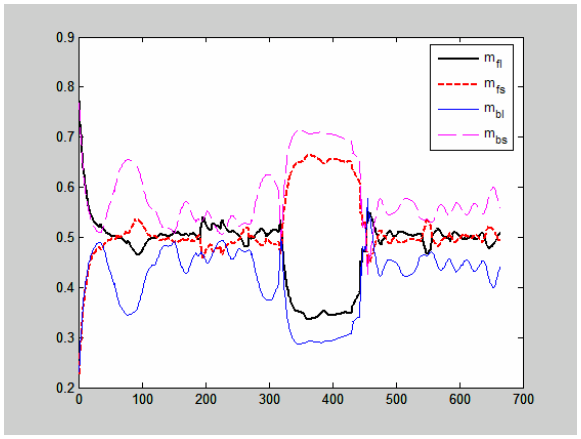

Figure 7. The mixing weights of the appearance model for the *office* sequence.

of the object short-term component dominates during the period from frame #320 to #450, which corresponds to the low light period. That is, when the light is off, the proposed algorithm keeps tracking the head region mainly using the short-term component. Once the light is turned on again, the weights of the long-term component regains its importance, as shown in the figure.

6 Conclusions

In this paper we presented a classification-based visual tracking framework. It has the benefits of being efficient, globally optimal in the region of interest, and robust to scale variations. An appearance modeling approach is also presented. Experiments demonstrated that PCI is both efficient and robust.

References

[1] D. Comaniciu, V. Ramesh and P. Meer, "Kernel-Based Object Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 5, pp.564–577, May 2003.

[2] H.-T. Chen and T.-L. Liu, "Trust-Region Methods for Real-Time Tracking," *IEEE Intl. Conf. on Computer Vision*, July 2001.

[3] P. Viola and M. Jones, "Robust Real-time Object Detection," *International Journal of Computer Vision*, Vol. 57, pp.137–154, May 2004.

[4] D. W. Scott, *Multivariate Density Estimation*, New York: Wiley, 1992.

[5] R. Collins, "Mean-shift Blob Tracking through Scale Space,", *Computer Vision and Pattern Recognition (CVPR'03)*, June 2003.

[6] J. Yang, W. Lu and A. Waibel, "Skin-Color Modeling and Adaption, ", Asian Conf. on Computer Vision, 1998.

[7] Y. Wu and T. S. Huang, "Color Tracking by Transductive Learning," *Computer Vision and Pattern Recognition*, June 2000.

[8] R. Collins and Y. Liu, "On-Line Selection of Discriminative Tracking Features," *IEEE Intl. Conf. on Computer Vision*, Oct. 2003.

[9] A. Jepson, D. Fleet and T. El-Maraghi, "Robust Online Appearance Models for Visual Tracking,", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 10, pp.1296–1311, Oct. 2003.

[10] T. Cover and J. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.

[11] B. Han and L. Davis, "Robust Observations for Object Tracking", *International Conference on Image Processing*, 2005.

[12] C. Shen, M. Brooks and A. Hengel, "Fast Global Kernel Density Mode Seeking with Application to Localisation and Tracking", *IEEE Intl. Conf. on Computer Vision*, 2005.

[13] K. Nummiaro, E. Koller-Meier and L. Gool, "An Adaptive Color-Based Particle Filter", *Image and Vision Computing*, Vol.21, No.1, pp.99–110, 2003.

[14] B. Han and L. Davis, "On-Line Density-Based Appearance Modeling for Object Tracking", *IEEE Intl. Conf. on Computer Vision*, 2005.