Multimed Tools Appl (2018) 77:30067–30088
https://doi.org/10.1007/s11042-018-6180-5

Single object tracking using particle filter framework and saliency-based weighted color histogram

Mai Thanh Nhat Truong[1] · **Myeongsuk Pak**[1] · **Sanghoon Kim**[1]

Received: 23 March 2017 / Revised: 4 April 2018 / Accepted: 22 May 2018 /
Published online: 4 June 2018

Abstract Despite many years of research, object tracking remains a challenging problem, not only because of the variety of object appearances, but also because of the complexity of surrounding environments. In this research, we present an algorithm for single object tracking using a particle filter framework and color histograms. Particle filters are iterative algorithms that perform predictions in each iteration using particles, which are samples drawn from a statistical distribution. Color histograms are embedded in these particles, and the distances between histograms are used to measure likelihood between targets and observations. One downside of color histograms is that they ignore spatial information, which may produce tracking failure when objects appear that are similar in color. To overcome this disadvantage, we propose a saliency-based weighting scheme for histogram calculation. Given an image region, first its saliency map is generated. Next, its histogram is calculated based on the generated saliency map. Pixels located in salient regions have higher weights than those in others, which helps preserve the spatial information. Experimental results showed the efficiency of the proposed appearance model in object tracking under various conditions.

Keywords Object tracking · Particle filter · Color histogram · Saliency map

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11042-018-6180-5) contains supplementary material, which is available to authorized users.

✉ Sanghoon Kim
kimsh@hknu.ac.kr

[1] Department of Electrical, Electronic, and Control Engineering, Hankyong National University, 327 Jungang-ro, Anseong-si, Gyeonggi-do, Republic of Korea

1 Introduction

Object tracking is a highly attractive topic in computer vision because of its extensive practical use, playing important roles in many vision-based applications. With the advance of image acquisition technologies [24] and methods [29], object tracking can be used in traffic surveillance [30] or safety monitoring systems [9]. Recently, object tracking has also been used in augmented reality [3], robot vision [15], and military purposes [21]. The aim of object tracking algorithms is to produce a trajectory record of objects of interest over time by locating their positions in image sequences.

However, despite several years of research and development, object tracking remains a challenging problem. Specifically, maintaining high accuracy in practice is difficult for object tracking techniques because of the inconsistency of target objects and their surrounding environments. Object appearances and illumination conditions may vary and non-target objects may cause occlusions. The complexity of a tracking task increases when surrounding areas become more diverse or when target objects vary considerably in appearance. Addressing these many difficulties simultaneously is a daunting task. Thus, object tracking methods are usually designed to solve these specific tracking problems [27].

Several state-of-the-art tracking methods have been proposed to address these problems. They can be categorized into two groups based on model construction, namely, generative and discriminative methods [28]. Discriminative methods consider tracking as a classification task. They try to discriminate the visual appearance of target objects from the background [4, 18]. By contrast, generative methods regard tracking as an optimization task. They search for regions that are most similar to the target object [16, 17]. Comprehensive reviews regarding object tracking are given in [27, 32, 33].

In this research, we use a particle filter as the main tracking framework. Particle filters are generative methods used to estimate system states in state-space models; these system states are observed over time. Particle filters solve the estimation problem by using sequential Monte Carlo sampling, in which a set of samples called particles is used to perform numerical approximation. Particle filters perform very well with nonlinear and non-Gaussian estimation problems, proving their efficiency in dealing with various difficulties related to object tracking [7].

In particle filter-based tracking algorithms, the appearance model of the target object is essential because it directly affects the properties of particles. Since the invention of the particle filter, several types of appearance models for this framework have been proposed, including color [19], contour [13], edge [10], and saliency [26]. Among these, color-based models are simple and effective. They have continued to be reliable even after several new appearance models were later proposed [22]. The color of an object is affected mainly by two physical elements: the spectral power distribution of the illuminant and the surface properties of the object [33]. For these reasons, color-based models can fully capture the appearance properties of target objects. However, a particle filter itself is a high complexity algorithm because each particle must be processed separately. Complex models can dramatically increase the overall execution time of a particle filter framework, rendering it useless in real-life applications. Therefore, not only effective but also simple models that can be constructed easily are necessary. Color-based models are candidates that satisfy these requirements.

Color-based models usually comprise a form of color distribution or color histogram. One flaw with color histograms is that they ignore the spatial layouts of targets, which can cause tracking failures when objects appear that are similar in color. To overcome this disadvantage, we propose a saliency-based weighting scheme for histogram calculation. In this

method, saliency maps of target objects and particles are generated. Their weighted histograms are then calculated based on these generated saliency maps. Pixels located in salient regions have higher weights than in others, which helps preserve the spatial information of image regions and increase the reliability of likelihood calculation.

There were several researches regarding object tracking that use saliency and color information [26, 28, 34, 35]. In these studies, saliency and color information are used separately to support each other. In our study, we introduce a simple but effective algorithm for object tracking. Saliency information is used to create a new type of histogram that can retain the details of spatial information. This study is divided into six parts. In Section 2, we present the particle filter framework for object tracking. The saliency map generation algorithm and weighted color histogram construction for observation likelihood calculation are described in Section 3. Section 4 provides experimental results and Section 5 discusses future work related to the proposed method. We conclude the study in Section 6.

2 Particle filter for object tracking

Particle filters are derived from the Bayesian filter [5]. Both are probabilistic methods that use noisy observations from a dynamic system to estimate the states of that system over time. This is possible because object tracking can be regarded as an estimation problem, in which the positions of objects in video frames are estimated from their observed positions in previous frames. Let $\mathbf{x}_t$ be the state of a given object (i.e., the center of the object region in video frames at time t). In general, the motion of an object can be expressed as a discrete dynamic system that takes the following form:

$$\mathbf{x}_t = f_{t-1}(\mathbf{x}_{t-1}, w_{t-1}), \tag{1}$$

$$\mathbf{z}_t = g_t(\mathbf{x}_t, \tilde{w}_t). \tag{2}$$

where f, g are the transition and measurement functions, respectively. System and measurement noises are denoted by w_t and $\tilde{w}_t$, respectively, and both have known distributions. These two distributions are usually Gaussian, but they are not necessarily identical.

At each time step t, the evolution $\mathbf{x}_t$ of the state $\mathbf{x}_{t-1}$ is calculated. Then, a series of observations $Z_t = \{\mathbf{z}_0, \ldots, \mathbf{z}_t\}$ is acquired. In practice, defining a dynamic system explicitly is difficult, however it can be estimated from observations. At a given time step k, Bayesian filter can recursively estimate the target state by performing state predictions and updates using the following two equations, respectively:

$$p(\mathbf{x}_k|Z_{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|Z_{k-1})\mathrm{d}\mathbf{x}_{k-1}, \tag{3}$$

$$p(\mathbf{x}_k|Z_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|Z_{k-1})}{p(\mathbf{z}_k|Z_{k-1})}, \tag{4}$$

where

$$p(\mathbf{z}_k|Z_{k-1}) = \int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|Z_{k-1})\mathrm{d}\mathbf{x}_k. \tag{5}$$

In other words, assuming that $p(\mathbf{x}_{k-1}|Z_{k-1})$ is known at time $k-1$, the prediction $p(\mathbf{x}_k|Z_{k-1})$ is calculated from the transition model $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and posterior density $p(\mathbf{x}_{k-1}|Z_{k-1})$. At time k, when the observation $\mathbf{z}_k$ becomes available, the prior probability density function $p(\mathbf{x}_k|Z_{k-1})$ can be updated, producing $p(\mathbf{x}_k|Z_k)$.

However, obtaining an explicit form of the integral operation in (3) is difficult. A particle filter solves this problem by using sequential Monte Carlo sampling, in which a set of samples, which are also called particles, is used to perform numerical approximation. In object tracking techniques based on particle filter frameworks, particles are usually in the form of rectangular windows associated with encoded features of the image regions inside those windows. Figure 1 illustrate a particle filter-based object tracking technique in action.

Let $S_t = \{\mathbf{x}_t^i, \omega_t^i \mid i = 1, \ldots, N\}$ be a set of N particles at time t. Each particle $\mathbf{x}_t^i$ is associated with a weight ω_t^i, where $\sum_{i=1}^{N} \omega_t^i = 1$ for all t. The particle filter algorithm is executed through iterations. At $t = 0$, S_0 is initialized by randomly assigning state and weight for each particle. When $t = k > 0$, at the beginning of the iteration, each particle advances independently based on a predefined transition model. This produces an approximation of the a priori probability density function as given by the following equation.

$$p(\mathbf{x}_k) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\mathbf{x}_k^i}(\mathbf{x}_k), \tag{6}$$

where δ denotes the Dirac's measure with support $\mathbf{x}_k^i$.

When the observation $\mathbf{z}_k$ is acquired from the dynamic system, the update step (4) is approximated using the following equation.

$$P(\mathbf{x}_k|Z_k) = \sum_{i=1}^{N} \omega_k^i \delta_{\mathbf{x}_k^i}(\mathbf{x}_k), \tag{7}$$

where

$$\omega_k^i \propto \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)}{\sum_{j=1}^{N} p(\mathbf{z}_k|\mathbf{x}_k^j)\omega_{k-1}^j} \omega_{k-1}^i. \tag{8}$$

As shown in (8), the weight of each particle is also updated based on likelihood with the target state. Particles that correspond to the most probable state will have a higher weight than others. The probability $p(\mathbf{z}_k|\mathbf{x}_k^i)$ is likelihood between the target state and the particles. This represents a critical part of the particle filter algorithm. In object tracking problems, the

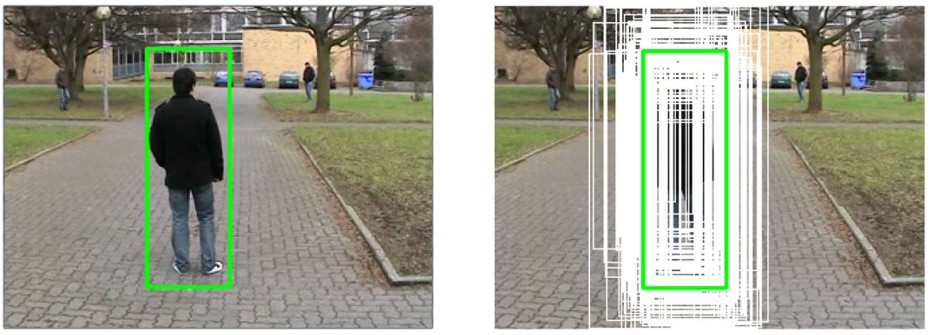


Fig. 1 Target object indicated by a green rectangle (left), and the particles at the current state indicated by white rectangles (right)

likelihood is usually measured by the distance between appearance models. The estimator $\hat{\mathbf{x}}_k$ for the state of the target is then calculated by:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{N} \omega_k^i \mathbf{x}_k^i. \tag{9}$$

At the end of each iteration, the particles are resampled, producing a new set of particles for the next iteration. Without resampling, the whole weight will likely be accumulated at a single particle after a few iterations. The new set of particles is generated by redistributing all current particles based on their current weights. Particles with higher weights are more likely to be selected. After being selected, each new particle is assigned a new weight, $1/N$. A review of resampling methods for particle filters is provided in [14].

The object tracking framework in this research is based on the condensation algorithm [7], a well-known implementation of a particle filter. For the transition model, we use the nearly constant velocity model [12]. The state transition equation is defined as:

$$\mathbf{x}_t = F_{t-1}\mathbf{x}_{t-1} + W_{t-1} \tag{10}$$

$$\mathbf{x}_{t-1} = [x_{t-1}\ y_{t-1}\ \dot{x}_{t-1}\ \dot{y}_{t-1}] \tag{11}$$

$$F_{t-1} = \begin{bmatrix} 1 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

where (x_{t-1}, y_{t-1}) is the location of the center of the target region at time $t-1$; $(\dot{x}_{t-1}, \dot{y}_{t-1})$ represents the motion of the state; Δ_t is the time interval between video frames; and W_{t-1} is the system noise, which has a Gaussian distribution. In the next section, we present the algorithm for observation likelihood calculation, which is used to update particle weights.

3 Observation likelihood

As previously mentioned, the particle filter-based object tracker requires a similarity measurement for the purpose of calculating the likelihood between particles and the target object. Particle filters are high complexity algorithms. Thus, a simple feature descriptor is required to maintain execution of the entire tracking algorithm at a reasonable time. For this reason, a color histogram model and saliency map are used, as they can be easily calculated. In our tracking method, the bounding box of the target object and particles are rectangular regions in the video frames, similar to Fig. 1. The likelihood between target object and particles is calculated as follows. First, the saliency-based weighting maps are extracted from the target and particles. Second, the weighted histograms of the target and particles are calculated using extracted weighting maps. Finally, Hellinger distance between the histograms of the target object and of each particle is computed. Higher distances mean lower likelihood. The details of each step are presented in the next two subsections.

3.1 Saliency-based weighting map extraction

A region is considered salient if its feature strength is stronger than that of its neighbors. Saliency detection techniques are methods that produce saliency maps from given image data. Several saliency detection techniques have been proposed [2], and can be classified as two types of approaches: bottom-up, which uses basic features such as colors or edges;

and top-down, which uses knowledge-driven properties. In our saliency-based weighting scheme for histograms, we use the spectral residual method from [6]. This saliency model is independent of features or prior knowledge. Moreover, the spectral residual method can construct saliency maps in a short amount of time, which is suitable for a particle filter-based object tracking method.

Given a rectangular image region $\mathcal{I}$ (of the target object or particles), a saliency map of that region is generated using the spectral residual method [6]. First the image $\mathcal{I}$ is down-sampled to 64×64 pixels, then its log spectrum is calculated as:

$$\mathcal{L}(\mathcal{I}) = \log(\Re(\mathfrak{F}[\mathcal{I}])) \tag{13}$$

where $\mathfrak{F}[\cdot]$ represents two-dimensional (2D) Fourier transform operation, and the log spectrum is a logarithm of the real part of the transformation result. The spectral residual $\mathcal{R}(\mathcal{I})$ of the image is defined as:

$$\mathcal{R}(\mathcal{I}) = \mathcal{L}(\mathcal{I}) - h_n * \mathcal{L}(\mathcal{I}) \tag{14}$$

where h_n is an $n \times n$ matrix defined by

$$h_n = \frac{1}{n^2} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{15}$$

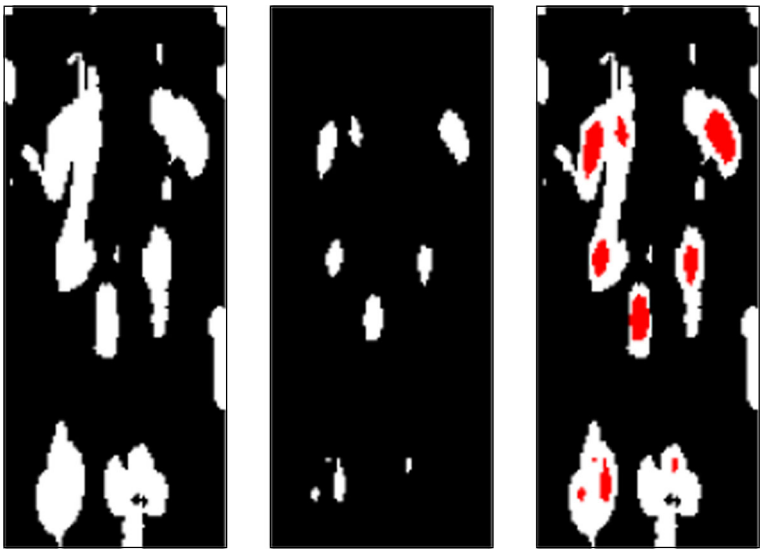
As suggested by the authors of this method, n equals 3. We then perform inverse 2D Fourier transform and smoothing using a Gaussian filter.

$$\mathcal{S}(\mathcal{I}) = \gamma(\cdot) * \mathfrak{F}^{-1}\left[\exp(\mathcal{R}(\mathcal{I}) + \mathcal{P}(\mathcal{I}))\right]^2 \tag{16}$$

where $\mathfrak{F}^{-1}$ represents the inverse 2D Fourier transform operation, $\gamma(\cdot)$ is a Gaussian smoothing kernel, and $\mathcal{P}(\mathcal{I}) = \Im(\mathfrak{F}[\mathcal{I}])$. Finally, the image $\mathcal{S}(\mathcal{I})$ is resized to its original, thereby producing the saliency map. The saliency map calculation is illustrated in Fig. 2.

This saliency map is then thresholded, producing two binary images. The thresholds are calculated by using the multi-level Otsu method [20]. The idea of this step is that, by using

Fig. 2 Image region (left) and its corresponding saliency map calculated using the spectral residual algorithm (right)



(a) Low-value saliency regions (b) High-value saliency regions (c) Weighting map

Fig. 3 Binarized saliency maps from Fig. 2 and its weighting map for histogram calculation

multi-level thresholding, we can retain multiple levels of saliency. Figure 3a and b present the binarized saliency maps of the image region shown in Fig. 2, and also indicate regions that pass the low and high thresholds, respectively. Finally, two binary images are combined to create the weighting map (Fig. 3c). This saliency-based weighting map is used in the next step to create the weighted histogram.

3.2 Saliency-based weighted color histogram

In addition to physical properties, color features also depend on color models, in which colors are represented in a mathematical manner as tuples of numbers. Several color models have been proposed for difference purposes, and red-green-blue (RGB) is the most commonly used color model [1]. Currently, no color model exists that is suitable for all vision-based applications. The selection of color models is based on the objectives of the application. In this research, the RGB model is selected because of its robustness against noise and occlusion. As such, the properties of particles are represented by three-dimensional RGB histograms. However, normal color histograms ignore the spatial layout of targets, which may lead to tracking failures when objects appear that are similar in color.

To overcome this disadvantage, we use the weighting map created previously (as illustrated in Fig. 3c). When we apply this weighting map to histogram construction, pixels in the red regions are counted thrice, those in the white regions are counted twice, and others are counted once. Therefore, given a pixel at location $\mathbf{l}$ in the rectangular region, the weighting function is defined as:

$$k(\mathbf{l}) = \begin{cases} 3 & \mathbf{l} \in \text{red regions} \\ 2 & \mathbf{l} \in \text{white regions} \\ 1 & \text{otherwise} \end{cases} \tag{17}$$

Let $h(\mathbf{l})$ be the function that assigns a given color at location $\mathbf{l}$ to one of the M bins of the histogram. The weighted 3D color histogram $P = \{P_u\}_{u=1...M}$ of an image region can be formulated as:

$$P_u = \sum_{i}^{I} k(\mathbf{l}_i)\delta[h(\mathbf{l}_i) - u] \tag{18}$$

where I is the number of pixels in the given image region and δ denotes the Kronecker function. In this research, the 3D histograms are constructed in RGB space using $8 \times 8 \times 8$ bins.

The saliency-based weighting function in (17) is applied for the purpose of retaining the spatial information of image regions. As shown in Fig. 2 and 3, the saliency map exposes the spatial structure of objects in an image region. The structure is then embedded into the weighted histogram. Therefore, the saliency-based weighting scheme can capture the spatial information of the image regions, which helps increase the reliability of the color-based appearance model.

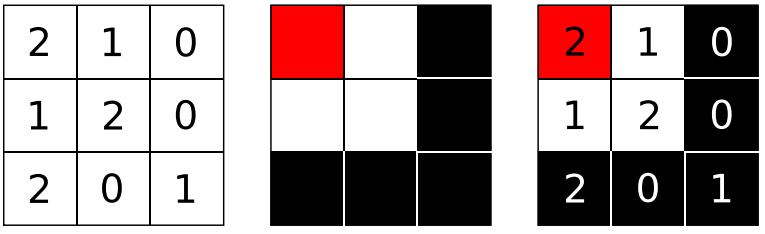
To illustrate the idea of this weighting scheme, we use a 3×3 image with three gray levels (Fig. 4). In a normal histogram (Fig. 5a), the pixel count for all gray levels is 3. In a weighted histogram (Fig. 5b), we have:

$$\begin{aligned} P_0 &= 1 \times 1 + 1 \times 1 + 1 \times 1 = 3 \\ P_1 &= 2 \times 1 + 2 \times 1 + 1 \times 1 = 5 \\ P_2 &= 3 \times 1 + 2 \times 1 + 1 \times 1 = 6 \end{aligned} \tag{19}$$

Note that in the above equations we only include operands where $\delta[h(\mathbf{l}_i) - u] \neq 0$, and the bins start from 0 to match the gray values. A comparison of the normal and weighted histograms is given in Fig. 5.

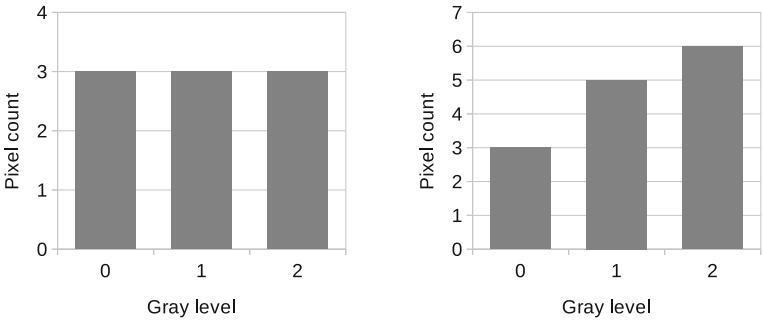
After obtaining the weighted histograms, we calculate the distances between the two color histograms using the Hellinger distance. This measure is also called the Bhattacharyya distance because it is derived from the Bhattacharya coefficient. The Hellinger distance is used because of its effectiveness in measuring differences between histograms [31]. Let $P = \{P_u\}_{u=1...M}$ and $Q = \{Q_u\}_{u=1...M}$ be the two weighted color histograms of the target object and a particle, respectively, the Bhattacharya coefficient of these two histograms is then given by:

$$B_C(P, Q) = \sum_{u=1}^{M} \sqrt{P_u Q_u}. \tag{20}$$



(a) Example 3-by-3 image (b) Weighting map (c) Weighting map applied to image

Fig. 4 Example using a 3×3 image with three gray levels



(a) Normal histogram (b) Saliency-based weighted histogram

Fig. 5 Comparison of two types of histogram calculated from Fig. 4

The Bhattacharyya coefficient is a measurement of the amount of overlap between two distributions. A higher coefficient value means that the two histograms are more similar to each other, which yields a shorter distance between them. The Hellinger distance is then defined as:

$$H_D(P, Q) = \sqrt{1 - B_C(P, Q)}. \tag{21}$$

Finally, the weights of particles are calculated as:

$$\omega \propto \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{H_D}{2\sigma^2}} \tag{22}$$

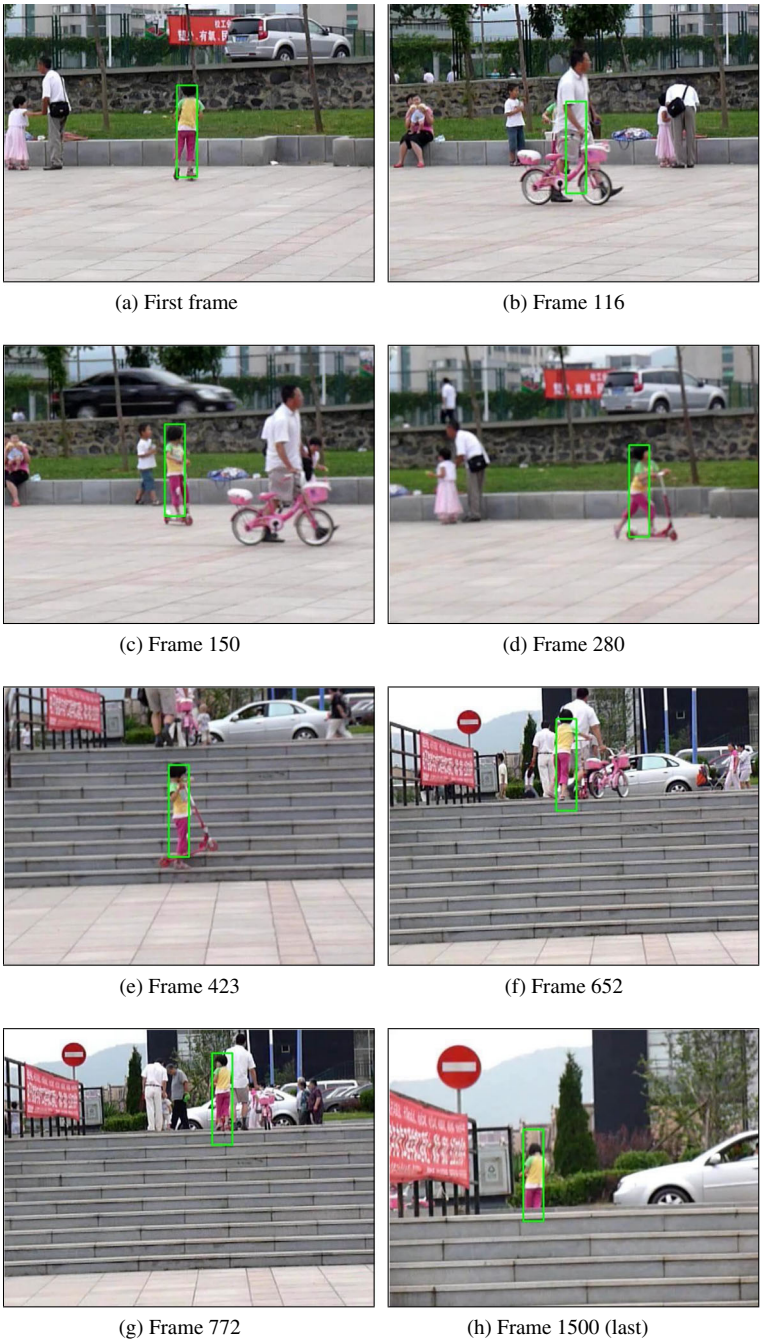
4 Experimental results

4.1 Performance evaluation

We carried out several experiments with different conditions to evaluate the performance of the particle filter-based object tracking framework and the proposed appearance model. The hardware platform for our experiments was a desktop computer equipped with a quad-core processor running at 3.0 GHz and 8 GB of RAM. The object tracking program was implemented in C++ on a Linux operating system. OpenCV library was used to process the video frames. We tested four video sequences that included various target appearances and environmental conditions. The four video sequences were entitled *Basketball*, *Bolt1*, *Girl*, and *Iceskater1*, all of which were obtained from [11]. In all experiments, the particle filter framework was configured for execution with 300 particles.

In the image sequence named *Girl*, the color patterns of the target region were distinct from the background. This helped our proposed appearance model fully capture the properties of the target, resulting in high tracking accuracy. As shown in Fig. 6, a full temporal occlusion occurred at Frame 116 and lasted for several frames. When the occlusion ended, the tracking algorithm successfully recovered, then continued producing high accuracy tracking results for the remainder of the image sequence despite several changes in posture of the image subjects.

In the subsequent test, we used an image sequence name *Iceskater1*, which showed an ice skater in action. Because of the nature of this sport, athletes in this sport typically



(a) First frame (b) Frame 116

(c) Frame 150 (d) Frame 280

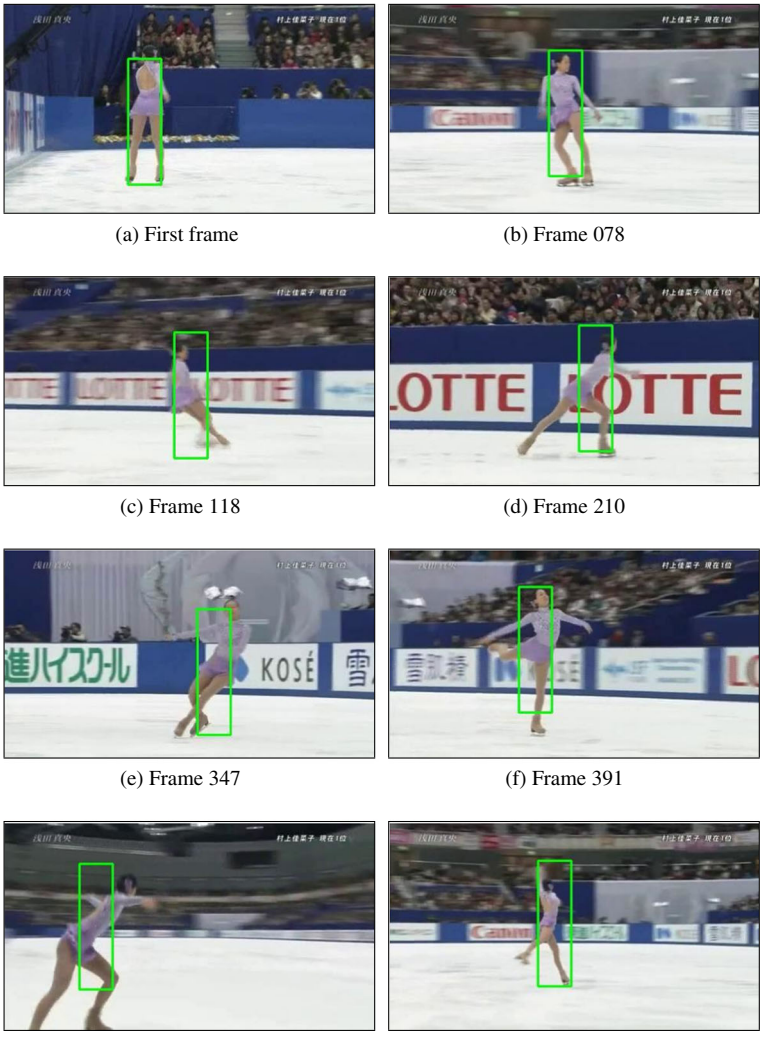
(e) Frame 423 (f) Frame 652

(g) Frame 772 (h) Frame 1500 (last)

Fig. 6 Tracking results from the sequence *Girl*

have fast and complex movements. As shown in Fig. 7, in addition to complex movements, the posture of the athlete constantly changes, as does the distance between the athlete and camera. We noted that the color of the athlete's outfit was rather similar to the background. However, the tracking algorithm produced accurate results until the last frame of the video.

Finally, we evaluated the ability of the proposed appearance model to distinguish similar regions. In this test, we used two image sequences: *Bolt1* and *Basketball*. The sequences in *Bolt1* included a group of sprinters (Fig. 8), and that in *Basketball* included a basketball match (Fig. 9). All athletes in both sequences had a similar appearance. Partial occlusions also occurred in these sequences several times because of the movements of non-target



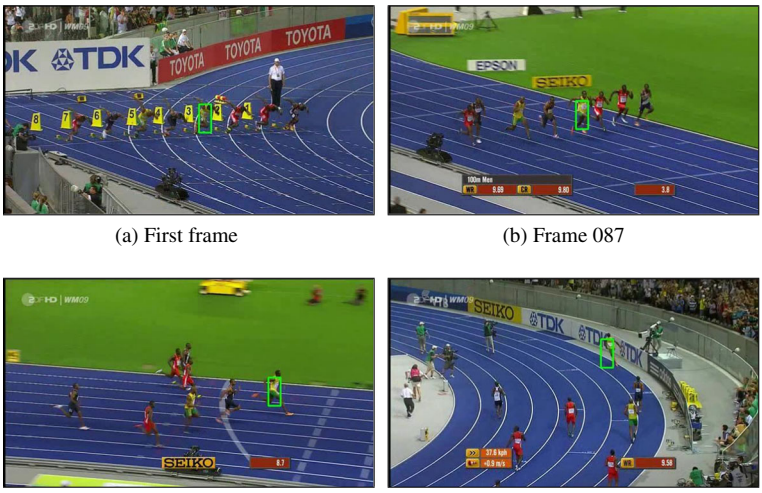
(a) First frame (b) Frame 078

(c) Frame 118 (d) Frame 210

(e) Frame 347 (f) Frame 391

(g) Frame 598 (h) Frame 661 (last)

Fig. 7 Tracking results from the sequence *Iceskater1*



(a) First frame (b) Frame 087

(c) Frame 207 (d) Frame 350 (last)

Fig. 8 Tracking results from the sequence *Bolt1*

athletes. In this test, the tracking algorithm successfully tracked target regions during the entire sequence for both *Bolt1* and *Basketball*.

For a quantitative analysis, we evaluated the accuracy of the proposed tracking method by calculating normalized center errors, which are normalized distances between the centers of tracking and ground-truth windows (Fig. 10). Let C_t and C_g be the positions of the

(a) First frame (b) Frame 471

(c) Frame 622 (d) Frame 725 (last)

Fig. 9 Tracking results from the sequence *Basketball*

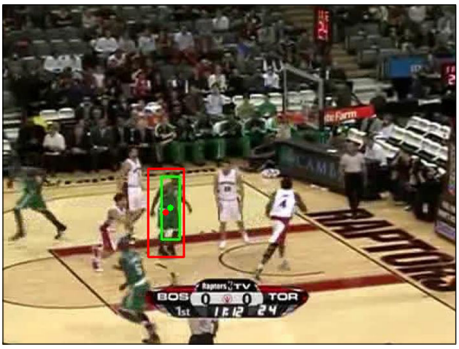


Fig. 10 Tracking window (green) and ground truth window (red) with their corresponding centers

centers of tracking and ground-truth windows, respectively. The normalized distance d_{norm} is calculated by:

$$d_{\text{norm}} = \frac{d(C_t, C_g)}{\sqrt{V_w^2 + V_h^2}} \tag{23}$$

where $d(C_t, C_g)$ is the Euclidean distance between C_t and C_g, V_w and V_h are the width and height of the video frame, respectively. As shown in Fig. 11, the errors of our tracking algorithm were less than 5% most of the time for all four tested sequences. Significant errors appeared in the sequence *Iceskater1* in which the maximum observed error was 12.25%. In this sequence, the target engaged in complex movements. However, high accuracy results were achieved most of the time in this sequence.

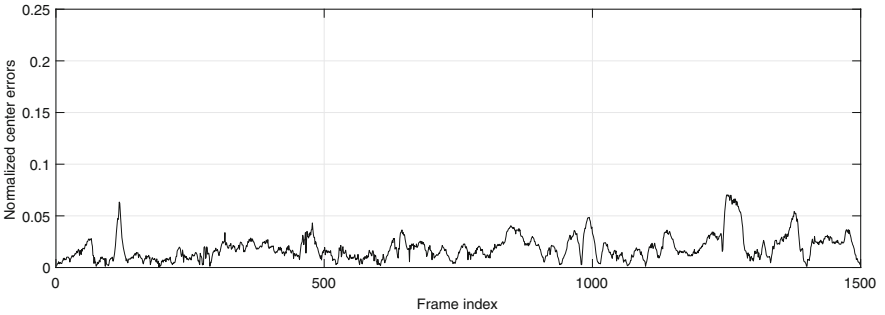
4.2 Comparison with other algorithms

For a performance comparison, we tested three other object tracking techniques: incremental learning tracking (IVT) [23], distribution field tracking (DFT) [25], and adaptive structural local sparse appearance model tracking (ASLSAM) [8]. Similarly to our approach, the authors of IVT and ASLSAM used a particle filter as a tracking framework with their proposed appearance models. The authors of DFT considered object tracking as an optimization problem.

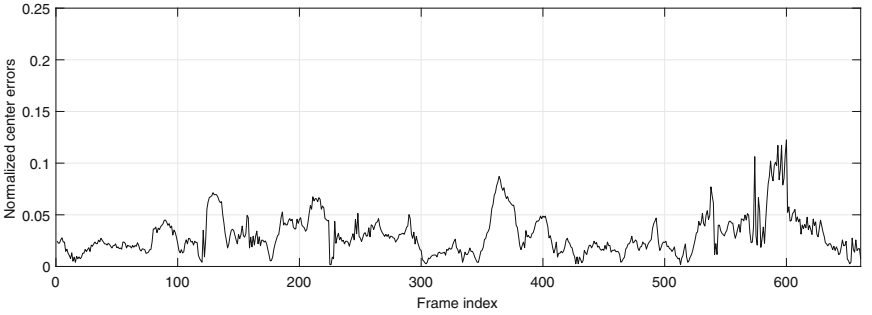
The implementation of the three aforementioned algorithms were acquired from authors' websites and configured for execution with default parameters. The image sequences used for this comparison were the same as those described in the previous section (*Basketball*, *Bolt1*, *Girl*, and *Iceskater1*). This experiment only covered tracking accuracy and did not consider the processing time of algorithms.

In the image sequence *Girl*, IVT lost track of the target after Frame 60, when the girl passed by another person (Fig. 12b). Because IVT only works with grayscale images, this might explain the failure, as the target and non-target objects had similar appearances in grayscale. DFT failed after Frame 111, when a short-time occlusion occurred and it could not subsequently recover (Fig. 12h). ASLSAM also failed after Frame 111 because of occlusion. However, it successfully recovered after the occlusion ended, then continued producing accurate results until the end of the image sequence (Fig. 12l).

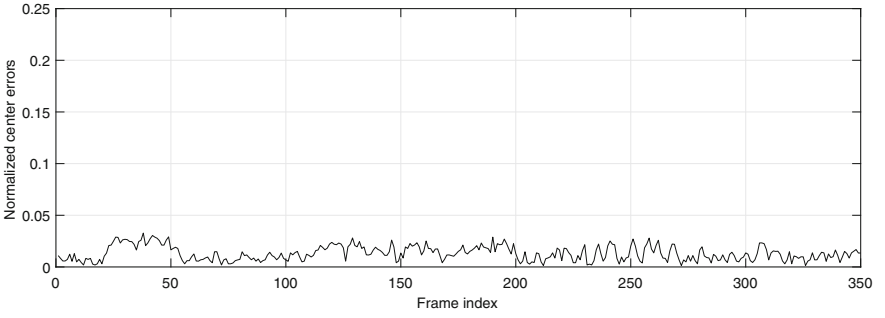
We used the sequence *Iceskater1* in the next test. As previously mentioned, the athlete in this image sequence had fast and complex movements. IVT failed after Frame 52, when the tracking window shrunk to the size of a dot and jumped around the video frames (Fig. 13c). DFT produced accurate tracking results in the first few seconds of the video, that is, until



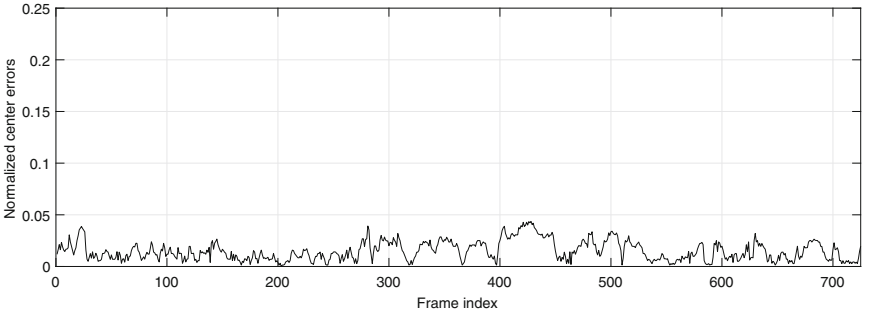
(a) Sequence *Girl*



(b) Sequence *Iceskater1*

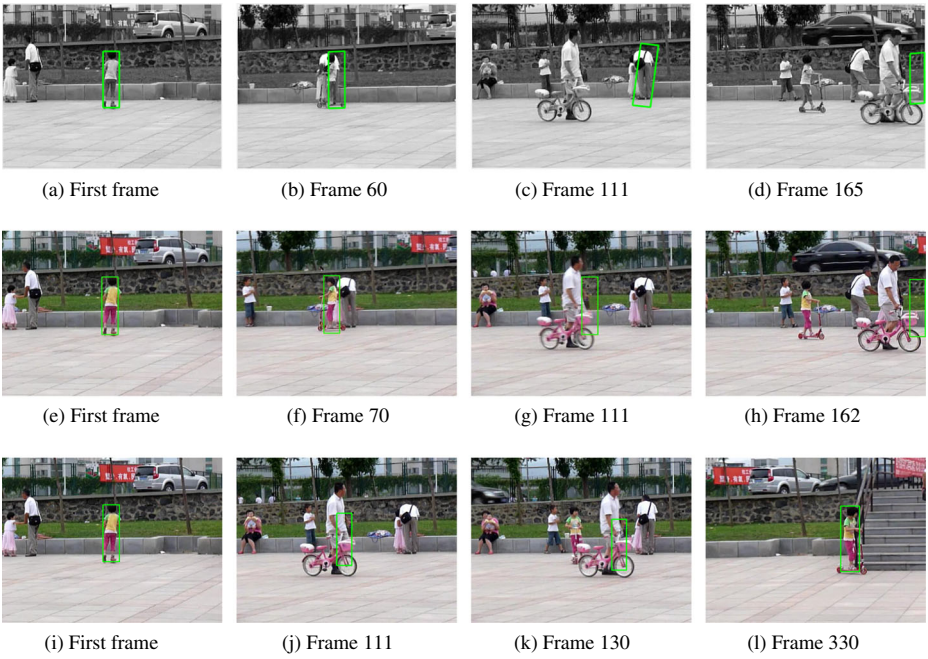


(c) Sequence *Bolt1*



(d) Sequence *Basketball*

Fig. 11 Normalized center errors from tested sequences



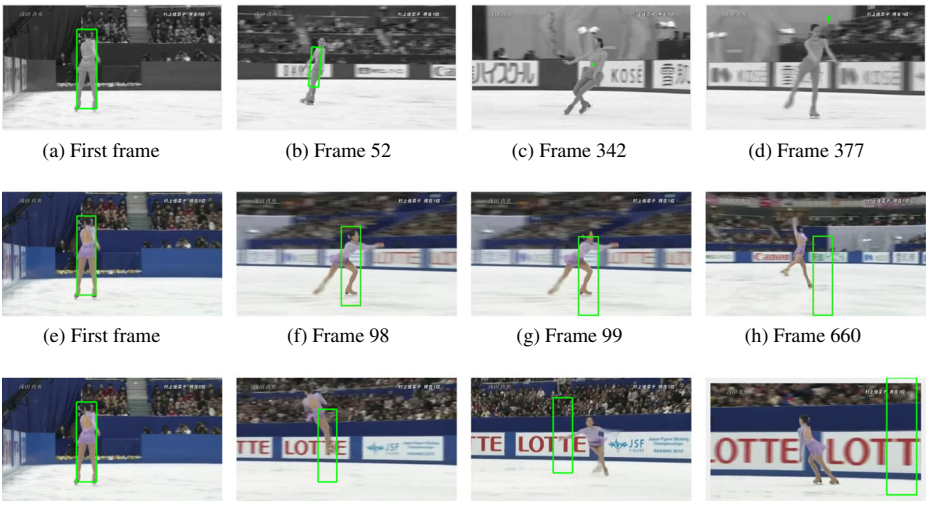
(a) First frame (b) Frame 60 (c) Frame 111 (d) Frame 165

(e) First frame (f) Frame 70 (g) Frame 111 (h) Frame 162

(i) First frame (j) Frame 111 (k) Frame 130 (l) Frame 330

Fig. 12 Tracking results from the sequence *Girl*. First row: IVT; second row: DFT; third row: ASLSAM

Frame 98. From Frame 99 and onward (Fig. 13g and h), the tracking window remained stuck at the bottom of the screen. ASLSAM also successfully tracked the target until Frame 130. At this point, the athlete jumped suddenly to perform a spin (Fig. 13j), ASLSAM could not follow this movement and was unable to recover from this tracking failure (Fig. 13k).



(a) First frame (b) Frame 52 (c) Frame 342 (d) Frame 377

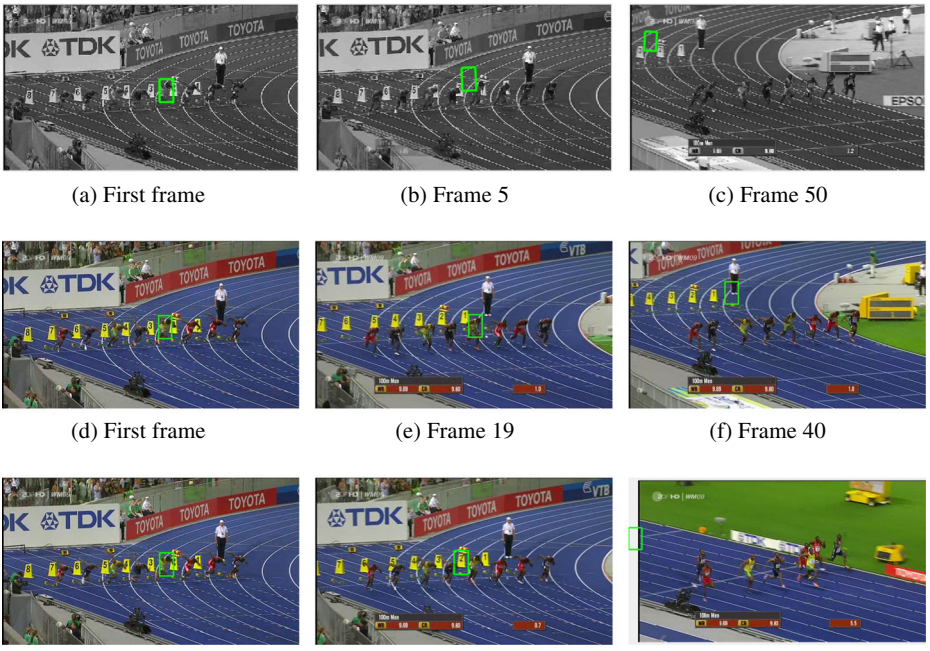
(e) First frame (f) Frame 98 (g) Frame 99 (h) Frame 660

(i) First frame (j) Frame 130 (k) Frame 150 (l) Frame 255

Fig. 13 Tracking results from the sequence *Iceskater1*. First row: IVT; second row: DFT; third row: ASLSAM

Finally, we evaluated the ability of the tracking algorithms to distinguish similar regions. In this test, two image sequences *Bolt1* and *Basketball* were used. In *Bolt1*, all three tracking algorithms failed after the first few frames of the sequence because the target object moved very fast, and several regions were present whose appearance was similar to that of the target object (Fig. 14). In the image sequence *Basketball*, all athletes had a similar appearance because of the team outfit. The tracking window of IVT shrunk to the size of a dot after a few frames. However, this small tracking window successfully tracked the target most of the time in this sequence (Fig. 15c). In the final seconds, the tracking window jumped to a non-target athlete that had a similar appearance as that of the target (Fig. 15d). DFT produced moderate results until Frame 79 (Fig. 15g), after which the tracking window remained stuck at the right edge of the video frame (Fig. 15h). ASLSAM also produced accurate tracking results. However, the tracking window could not handle partial occlusion between two similar objects (Fig. 15k and l), and then the tracking window jumped to a non-target object.

For quantitative analysis, we compared normalized center errors of our proposed method with those of IVT, DFT, and ASLSAM. As shown in Fig. 16, the errors of our tracking algorithm are significantly lower than those of other methods. For the sequence *Girl*, our proposed method produced highly accurate results during the course of the entire video, ASLSAM successfully recovered from full occlusion but considerable time was spent in this recovery. IVT and DFT produced high error rates. For the sequence *Iceskater1*, the center errors of all four algorithms varied significantly because of the complex movements of the target. The center errors from our algorithm was low, whereas other algorithms produced low accuracy and unstable results. For the sequence *Bolt1*, our method succeeded in tracking the sprinter, but other methods failed after the first few frames. For the last sequence, *Basketball*, DFT failed quite early, IVT and ASLSAM produced accurate results

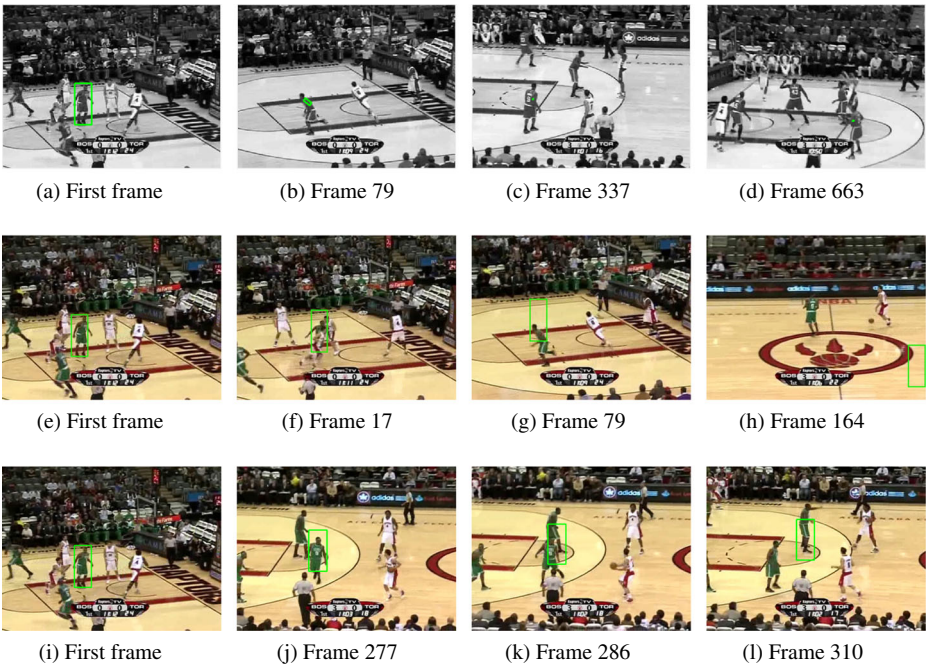


(a) First frame (b) Frame 5 (c) Frame 50

(d) First frame (e) Frame 19 (f) Frame 40

(g) First frame (h) Frame 13 (i) Frame 130

Fig. 14 Tracking results from the sequence *Bolt1*. First row: IVT; second row: DFT; third row: ASLSAM



(a) First frame (b) Frame 79 (c) Frame 337 (d) Frame 663

(e) First frame (f) Frame 17 (g) Frame 79 (h) Frame 164

(i) First frame (j) Frame 277 (k) Frame 286 (l) Frame 310

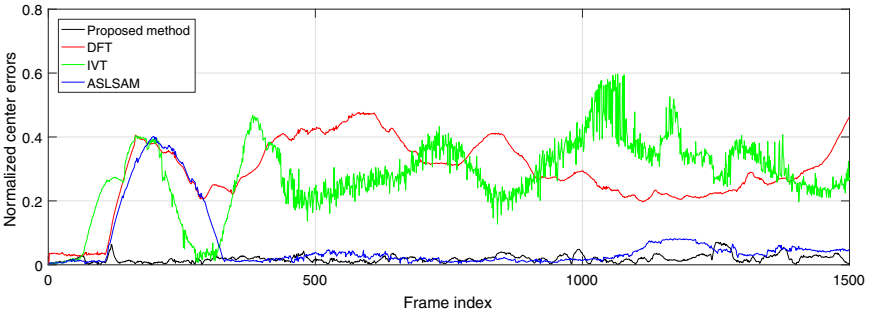
Fig. 15 Tracking results from the sequence *Basketball*. First row: IVT; second row: DFT; third row: ASLSAM

until occlusions between similar objects appeared and the two methods subsequently failed to recover. Our method successfully tracked the target until the end of the video.

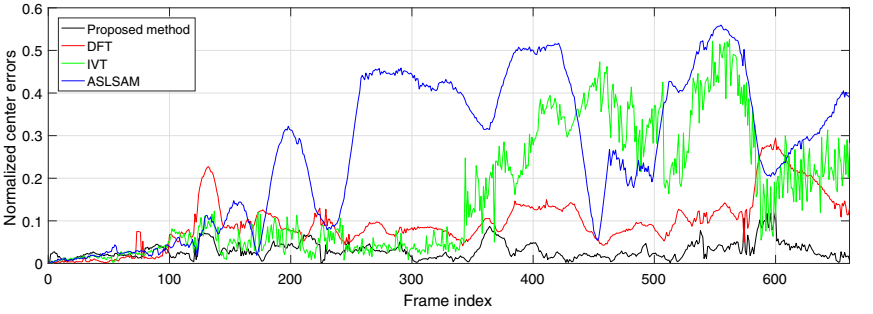
5 Discussion and future work

Although the proposed particle object tracking algorithm proved its efficiency and effectiveness after being subjected to several tests, we cannot guarantee that the proposed algorithm will always be accurate in all situations. Figure 17 shows the tracking results for the sequence *Pedestrian2*, which was also acquired from [11]. The target was tracked accurately until Frame 170. After this frame, the camera moved abruptly for a short distance (Frame 181). The dynamic model was unable to follow this sudden change, and the tracking algorithm could not recover from its original failure because of a nearby non-target object that had a similar appearance. Moreover, the area of the target region was small compared to the size of the video frame. Our model failed to retain the spatial structure of the target region when the area was insufficiently large.

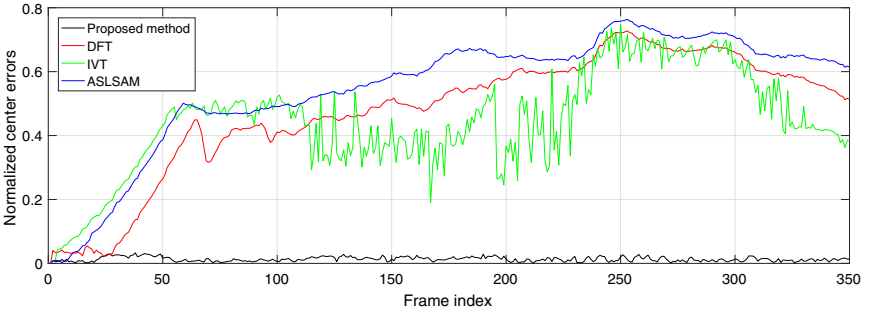
The proposed appearance model relies on two components: a saliency model and a thresholding method. Different algorithms produce different results, which in turn affect the weighting scheme considerably. Appropriate saliency and thresholding methods must retain the spatial information as much as possible while having a reasonable execution time. For a future study, we will exploit modern techniques to produce better saliency maps and more efficient thresholding algorithm to retaining the details of spatial information, improving the reliability of our appearance model.



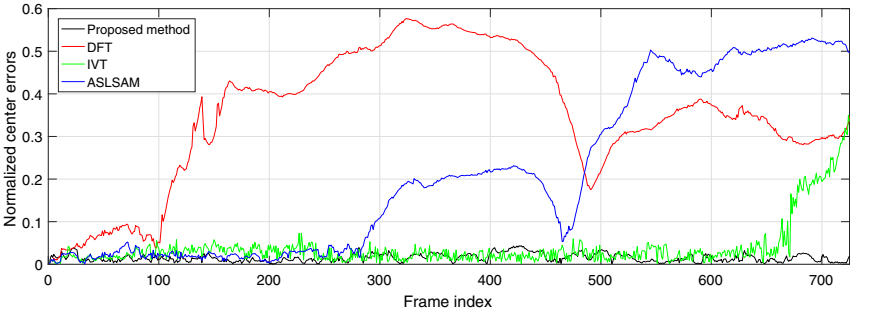
(a) Sequence *Girl*



(b) Sequence *Iceskater1*



(c) Sequence *Bolt1*



(d) Sequence *Basketball*

Fig. 16 Comparison of normalized center errors from tested sequences

(a) Frame 170 (b) Frame 181 (c) Frame 189 (d) Frame 269

Fig. 17 Tracking results from the sequence *Pedestrian2*

6 Conclusion

In this study, we proposed a novel appearance model for a particle filter-based object tracking method. Color-based features are suitable candidates for high complexity tracking frameworks such as particle filters. These features are usually in the form of histograms, which are simple to construct. One downside of color histograms is that they ignore the spatial layouts of targets, which may cause tracking failures when objects appear that are similar in color. To overcome this disadvantage, we proposed a saliency-based weighting scheme for histogram calculation. Given an image region, its corresponding saliency map is first generated. Then, its weighted histogram is calculated based on the generated saliency map. Pixels located in salient regions have higher weight than in others, which helps preserve the spatial information of image regions. The color histograms are then embedded in particles and the distances between histograms are used to measure observation likelihood. Experimental results showed the efficiency of the proposed appearance model in object tracking under various conditions. For future work, we will improve the performance of our model by producing better saliency maps and more efficient thresholding algorithm to retaining the details of spatial information.

Funding This study was funded by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2015R1D1A1A01057518).

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interests.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Barthélemy Q, Larue A, Mars JI (2015) Color sparse representations for image processing: review, models, and prospects. IEEE Trans Image Process 24(11):3978–3989. https://doi.org/10.1109/TIP.2015.2458175
2. Borji A, Itti L (2013) State-of-the-art in visual attention modeling. IEEE Trans Pattern Anal Mach Intell 35(1):185–207. https://doi.org/10.1109/TPAMI.2012.89

3. Bostanci E, Kanwal N, Clark AF (2015) Augmented reality applications for cultural heritage using Kinect. Human-Centric Comput Informa Sci 5(1):1–18. https://doi.org/10.1186/s13673-015-0040-3
4. Dinh TB, Yu Q, Medioni G (2014) Co-trained generative and discriminative trackers with cascade particle filter. Comput Vis Image Underst 119:41–56. https://doi.org/10.1016/j.cviu.2013.11.003
5. Haug AJ (2012) Bayesian estimation and tracking: a practical guide. Wiley
6. Hou X, Zhang L (2007) Saliency detection: a spectral residual approach. In: 2007 IEEE Conference on computer vision and pattern recognition, pp 1–8. https://doi.org/10.1109/CVPR.2007.383267
7. Isard M, Blake A (1998) Condensation-conditional density propagation for visual tracking. Int J Comput Vis 29(1):5–28. https://doi.org/10.1023/A:1008078328650
8. Jia X, Lu H, Yang MH (2012) Visual tracking via adaptive structural local sparse appearance model. In: 2012 IEEE Conference on computer vision and pattern recognition, pp 1822–1829. https://doi.org/10.1109/CVPR.2012.6247880
9. Juhyun L, Hanbyul C, Kicheon H (2015) A fainting condition detection system using thermal imaging cameras based object tracking algorithm. J Converg 6(3):1–15
10. Klein G, Murray DW (2006) Full-3d edge tracking with a particle filter. In: Proceedings of the British machine vision conference, pp 114.1–114.10. https://doi.org/10.5244/C.20.114
11. Kristan M, Matas J, Leonardis A, Vojíř T, Pflugfelder R, Fernández G, Nebehay G, Porikli F, Čehovin L (2016) A novel performance evaluation methodology for single-target trackers. IEEE Trans Pattern Anal Mach Intell 38(11):2137–2155. https://doi.org/10.1109/TPAMI.2016.2516982
12. Li XR, Jilkov VP (2003) Survey of maneuvering target tracking. Part I. dynamic models. IEEE Trans Aerosp Electron Syst 39(4):1333–1364. https://doi.org/10.1109/TAES.2003.1261132
13. Li P, Zhang T, Pece AE (2003) Visual contour tracking based on particle filters. Image Vis Comput 21(1):111–123. https://doi.org/10.1016/S0262-8856(02)00133-6
14. Li T, Bolic M, Djuric PM (2015) Resampling methods for particle filtering: classification, implementation, and strategies. IEEE Signal Process Mag 32(3):70–86. https://doi.org/10.1109/MSP.2014.2330626
15. Lin S, Garratt MA, Lambert AJ (2017) Monocular vision-based real-time target recognition and tracking for autonomously landing an UAV in a cluttered shipboard environment. Auton Robot 41(4):881–901. https://doi.org/10.1007/s10514-016-9564-2
16. Liu B, Yang L, Huang J, Meer P, Gong L, Kulikowski C (2010) Robust and fast collaborative tracking with two stage sparse optimization. In: Proceedings of the 11th European conference on computer vision: part IV, pp 624–637
17. Mei X, Ling H (2009) Robust visual tracking using ℓ_1 minimization. In: 2009 IEEE 12th International conference on computer vision, pp 1436–1443. https://doi.org/10.1109/ICCV.2009.5459292
18. Meshgi K, Oba S, Ishii S (2016) Robust discriminative tracking via query-by-bagging. In: 2016 13th IEEE International conference on advanced video and signal based surveillance (AVSS), pp 8–14. https://doi.org/10.1109/AVSS.2016.7738027
19. Nummiaro K, Koller-Meier E, Gool LJV (2002) Object tracking with an adaptive color-based particle filter. In: Pattern Recognition, 24th DAGM symposium, Zurich, Switzerland, September 16-18, 2002, proceedings, pp 353–360. https://doi.org/10.1007/3-540-45783-6_43
20. Otsu N (1979) A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern 9(1):62–66. https://doi.org/10.1109/TSMC.1979.4310076
21. Pham I, Polasek M (2014) Algorithm for military object detection using image data. In: 2014 IEEE/AIAA 33rd Digital avionics systems conference (DASC), pp 3D3–1–3D3–15. https://doi.org/10.1109/DASC.2014.6979457
22. Possegger H, Mauthner T, Bischof H (2015) In defense of color-based model-free tracking. In: 2015 IEEE Conference on computer vision and pattern recognition (CVPR), pp 2113–2120. https://doi.org/10.1109/CVPR.2015.7298823
23. Ross DA, Lim J, Lin RS, Yang MH (2008) Incremental learning for robust visual tracking. Int J Comput Vis 77(1):125–141. https://doi.org/10.1007/s11263-007-0075-7
24. Sarif BA, Pourazad MT, Nasiopoulos P, Leung VC, Mohamed A (2015) Fairness scheme for energy efficient H.264/AVC-based video sensor network. Human-Centric Comput Inform Sci 5(1):7. https://doi.org/10.1186/s13673-015-0025-2
25. Sevilla-Lara L, Learned-Miller E (2012) Distribution fields for tracking. In: 2012 IEEE Conference on computer vision and pattern recognition, pp 1910–1917. https://doi.org/10.1109/CVPR.2012.6247891
26. Sidibé D, Fofi D, Mériaudeau F (2010) Using visual saliency for object tracking with particle filters. In: 2010 18th European signal processing conference, pp 1776–1780
27. Smeulders AWM, Chu DM, Cucchiara R, Calderara S, Dehghan A, Shah M (2014) Visual tracking: an experimental survey. IEEE Trans Pattern Anal Mach Intell 36(7):1442–1468. https://doi.org/10.1109/TPAMI.2013.230

28. Su Y, Zhao Q, Zhao L, Gu D (2014) Abrupt motion tracking using a visual saliency embedded particle filter. Pattern Recogn 47(5):1826–1834. https://doi.org/10.1016/j.patcog.2013.11.028
29. Sung Y, Kwak J, Park JH (2015) Graph-based motor primitive generation framework. Human-Centric Comput Inform Sci 5(1):35. https://doi.org/10.1186/s13673-015-0051-0
30. Vega-Maldonado S, Wario F, Arámburo-Lizárraga J, Perez-Cisneros M, Cedano-Olvera M (2015) Visual registration and tracking for traffic monitoring. In: 2015 IEEE First international smart cities conference (ISC2), pp 1–6. https://doi.org/10.1109/ISC2.2015.7366216
31. Vojir T, Noskova J, Matas J (2014) Robust scale-adaptive mean-shift for tracking. Pattern Recogn Lett 49:250–258. https://doi.org/10.1016/j.patrec.2014.03.025
32. Yang H, Shao L, Zheng F, Wang L, Song Z (2011) Recent advances and trends in visual tracking: a review. Neurocomputing 74(18):3823–3831. https://doi.org/10.1016/j.neucom.2011.07.024
33. Yilmaz A, Javed O, Shah M (2006) Object tracking: a survey. ACM Comput Surv 38(4):1–45. https://doi.org/10.1145/1177352.1177355
34. Yoo S, Kim W, Kim C (2014) Saliency combined particle filtering for aircraft tracking. J Signal Process Syst 76(1):19–31. https://doi.org/10.1007/s11265-013-0803-x
35. Yuan Y, Gao C, Liu Q, Wang J, Zhang C (2014) Using local saliency for object tracking with particle filters. In: 2014 IEEE International Conference on signal processing, communications and computing (ICSPCC), pp 388–393. https://doi.org/10.1109/ICSPCC.2014.6986221

Mai Thanh Nhat Truong received his B.Sc. degree in Mathematics and Computer Science from Ho Chi Minh City University of Science, Vietnam National University in 2014. After finishing undergraduate course, he worked as a teaching assistant until September 2015. Since then he has been studying master course at Department of Electrical, Electronic and Control Engineering, Hankyong National University, Anseong, Korea. His research interests are image processing and image understanding.

Myeongsuk Pak is a PhD Student in the school of Electrical, Electronic and Control Engineering at Hankyong National University, Anseong, Korea. She received her M.Sc. degree from Hankyong National University in 2016. Her research interests include image processing.

Sanghoon Kim was born in Seoul, Korea, in 1964. He received the B.Sc., M.Sc., and Ph.D. degrees in Electronic Engineering from Korea University, Seoul, in 1987, 1989, and 1999, respectively. From 1989 to 1994, he was a Research Engineer with LG Semiconductor Company, where he was engaged in research and development of PC chipset design. From January 2004 to January 2005, he was a Visiting Scholar with the University of Maryland, College Park, MD, USA. Since September 1999, he has been with Hankyong National University, Anseong, Korea, where he is currently a Professor. His current research interests are in the areas of image processing, object detection, and robot vision. Prof. Kim is a member of the IEEE and Korean Information Processing Society.