Trans Edutainment manuscript No.
(will be inserted by the editor)

Multiple-Cue-Based Visual Object Contour Tracking with Incremental Learning

Aiping Wang · **Zhiquan Cheng** ·
Ralph R. Martin · **Sikun Li**

the date of receipt and acceptance should be inserted later

Abstract This paper proposes a visual object contour tracking algorithm using a multi-cue fusion particle filter. A novel contour evolution energy is proposed which integrates an incrementally learnt model of object appearance with a parametric snake model. This energy function is combined with a mixed cascade particle filter tracking algorithm which fuses multiple observation models for object contour tracking. Bending energy due to contour evolution is modelled using a thin plate spline (TPS). Multiple order graph matching is performed between contours in consecutive frames. Both of the above are taken as observation models for contour deformation; these models are fused efficiently using a mixed cascade sampling process. The dynamic model used in our tracking method is further improved by the use of optical flow. Experiments on real videos show that our approach provides high performance object contour tracking.

Keywords Tracking · Discriminative methods · Snake model · Particle filter · Mixed cascade

1 Introduction

Visual object contour tracking is a fundamental problem in computer vision, and has been widely investigated because of its usefulness in many fields, such as video surveillance, object recognition, 3D reconstruction, and medical diagnosis.

This work was supported by the National Science Foundation of China(No.60873120, No.60970094, No.61003075)

A. Wang, Z. Cheng, S. Li
School of Computer, National University of Defense Technology, Changsha, China
E-mail: ipwang@nudt.edu.cn, cheng.zhiquan@gmail.com, lisikun@263.net.cn

R. R. Martin
School of Computer Science & Informatics, Cardiff University, Cardiff, UK
E-mail: Ralph.Martin@cs.cardiff.ac.uk

Active contour, or snake, models [11] are a popular approach to contour tracking. The basic idea is to drive evolution by minimizing energy functional for the object contour; this energy depends on internal spline forces and external image forces. Numerous active contour algorithms have been proposed; both parametric active contours [20, 2, 10, 23, 8, 17] and geometric active contours [4, 19, 16], are used, with different representations for the contour curve. In the former, the contour is approximated by an explicit parametric model, typically using a set of control points [11, 20, 23]; B-splines are often used [2, 10]. In the second case, the contour is typically represented by an implicit function, as in the level set method [19, 16]. In general, parametric contour methods are more efficient, and are thus more suitable for contour tracking in real-time.

Traditional snake models suffer from a serious limitation when used for tracking in image sequences: the convergence of results is very sensitive to the initial location of the contour. To deal with this problem, various estimation tools, such as the Kalman filter and particle filters, can be used to update parameter values over the sequence. For example, [20, 17, 7, 5] used a Kalman filter to track a fixed number of marker points, or parametric values, such as a B-spline's control points. However, the Kalman filter assumes linear system and measurement models, which is unsuitable for many applications.

In order to track contours with non-Gaussian and nonlinear state densities in cluttered video sequence, Isard and Blake [10] introduced the condensation algorithm. They used a B-spline representation for object contours, and particle filters to track the curve parameters given noisy observations. Since their approach only allows affine deformations of the contour, it is unsuitable for deforming objects, undergoing local deformations. Rathi et al. [18] combined a particle filtering algorithm with the geometric active contour framework to give an approach that can be used for tracking moving and deforming objects. However, they directly track affine deformations, while using an approximately linear observer to estimate any non-affine deformation of the object contour. Thus, their method cannot deal with complex contour deformations. Vaswani et al. [15] proposed a further algorithm, *Deform PF-MT*. They suggest that in most real problems, much of the contour deformation depends on a few parameters, while the deformation in the rest of the state space is small. Hence they use the deformations at a small sub-sampled set of locations along the contour as an effective basis space for particle filtering. However, they still explicitly track the contour deformations. In the presence of complexity and uncertainty of object deformations, their method is error-prone. Furthermore, the above approaches only use simple observation models, which which do not provide stable tracking target in the presence of large shape changes or significant occlusion.

Recently, discriminative learning methods, especially incremental ones, have received much attention; they aim to find a decision boundary that can best separate the object from the background, instead of just building a model which only describes the target. Incremental learning methods have been shown to be suitable for tracking objects with appearance variations or within cluttered environments. Wei et al. [22] integrated discriminative methods into a level set framework. However, this approach uses a set of off-line-trained weak classifiers. These cannot cope well with variations in object appearance, or noisy backgrounds. Furthermore, the tracking framework used simply builds on the detection results output by the classifiers, so is unstable due to the accumulation of classification errors.

In this paper, we propose a multi-cue based discriminative object contour tracking algorithm with two major contributions as follows.

1. We give a mixed cascade particle filter tracking algorithm using multiple observation models to improve the accuracy and stability of object contour tracking.
   The *incremental extremely random forest classifier* (IERF) [21] is adopted as an incremental learning approach to modelling target appearance. We describe contour deformation using separate inter-frame and intra-frame contour deformation models. The former is defined by using bending energy based on the thin plate spline (TPS) model [9], and multiple order graph matching [1] between contours in consecutive frames. The intra-frame deformation model is based on an energy which depends on the contour evolution process within the current frame.
   To fuse these observations efficiently, a *mixed cascade* importance sampling process is used. Our method combines a series of observers in multiple stages of importance sampling. However, we do not use a simple cascade. We allow multiple observer outputs to be blended during a single stage of the importance sampling process, and then combine multiple stages in cascade mode. Therefore, we refer to our method as *mixed cascade importance sampling*.
   We use optical flow information to further improve the accuracy of the particle filter model.
2. We give a novel contour evolution energy which integrates an incremental learning discriminative model with a parametric snake, giving improved performance in contour evolution.

Results on challenging video sequences demonstrate the effectiveness and robustness of our method.

The rest of the paper is organized as follows: Section 2 describes the IERF classifier for object tracking, our discriminative parametric snake model is presented in Section 3. The multi-cue mixed cascade particle filter tracking algorithm is discussed in Section 4. Experimental results are given in Section 5 and conclusions in Section 6.

2 Incremental Extremely Random Forests

First we briefly discuss the incremental learning discriminative model used in the paper: the *incremental extremely random forest* (IERF), originally introduced in [21]. The IERF classifier is suitable for online learning and classification of streaming data, and is especially useful in object tracking problems.

Extremely randomized trees provide a tree-based ensemble method for supervised classification. However it has to be trained in off-line mode: the entire training data must be given in advance. Using an incremental extremely random forest (IERF) leads to a tree-based ensemble method which can deal with online learning with streaming data.

An IERF classifier $H(\mathbf{x})$ builds an ensemble of decision trees, $H(\mathbf{x}) = \{h_i(\mathbf{x})\}_{i=1}^{N}$, where $\mathbf{x}$ is a sample with unknown label, and N is the total number of decision trees.

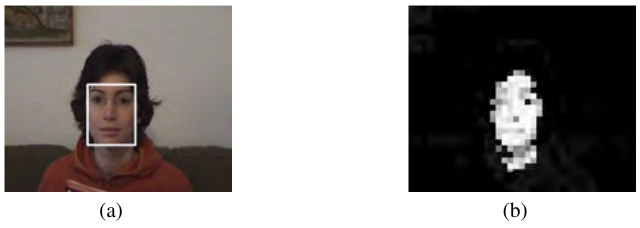

(a) (b)

Fig. 1 (a): target within the white rectangle, (b): confidence map provided by IERF classifiers.

Each decision tree is grown according to the classical top-down procedure without sample replacement. For each leaf node, an example list is maintained in memory and the a count is kept of the number of items in each labeled class derived from the incoming examples. When a new labeled example arrives, it is routed down the current tree to a leaf node, based on its attribute values. If certain splitting conditions are satisfied, then the leaf node is split into two, and the tree extended.

Let $c(\mathbf{x}) \in C = \{1, 2, \cdots, M\}$ be the true label for sample $\mathbf{x}$, where there are M possible labels. Each tree $h_i(\mathbf{x})$ outputs a value $\hat{c}_i(\mathbf{x})$ from label set C as its proposed classification for $\mathbf{x}$. The final output of $H(\mathbf{x})$ is the mode of $\{\hat{c}_i(\mathbf{x})\}_{i=1}^{N}$.

Provision of training data for online learning is an important issue for IERF classifiers. A typical solution is co-training [4], a commonly used approach in semi-supervised learning. The basic idea is that two classifiers classify the unlabeled data, and use this newly labeled data to update each other.

An IERF classifier can be easily applied to object tracking by integrating it into the co-training framework [21]. In this paper we use two IERF classifiers working separately in the color histogram space and the space of histograms of oriented gradients (HoG) [6]. In the first frame, the target is identified and a number of overlapping square subwindows are extracted at random positions from the target region and the background area. Then the two classifiers are initialized by using these few pieces of training data. In the following frames, both classifiers classify the incoming data, and and the co-training framework is used: the color classifier chooses some most confident samples to update the HoG classifier, and vice-versa.

The outputs from each IERF classifier are expressed as two independent confidence maps for the pixels, where the classification margin is used as the confidence measure. Large values in the confidence map are likely to belong to the target. We add these two confidence maps together to create a final confidence map: see Fig. 1.

As Fig. 1 shows, this tracking method using IERF classifiers can coarsely locate the target, narrowing it down to a small region of the original video frame. We take advantage of this characteristic to design a novel parametric snake model, which converges more accurately than an ordinary parametric snake model, as we explain in the next section.

Note that the IERF classifiers used in our tracking method can update themselves using the constraint of the object contour determined in each frame, which makes the tracking robust.

3 Discriminative parametric snake model

A *parametric* snake model is used for efficiency. Our model represents a curve defined by a set of discrete contour points (*snaxels*) $\{\mathbf{v}_i\}_{i=1}^{N_s}$, where N_s is the number of contour points and $\mathbf{v}_i = [x_i, y_i]^T$, the coordinates of $\mathbf{v}_i$ in the image. The snaxels are connected by line segments to create the curve. Then, following the usual approach, the contour evolution energy E_{snake} is represented as

$$E_{\text{snake}} = \sum_{i=1}^{N_s} \left(E_{\text{int}}(\mathbf{v}_i) + E_{\text{ext}}(\mathbf{v}_i)\right) \tag{1}$$

where E_{int} is the *internal energy* produces forces to make the snake contour smooth, while E_{ext} is the *external energy* produces forces attracting the snake to desired image features such as lines and edges.

The internal energy E_{int} at each snaxel is typically defined as $E_{\text{int}}(\mathbf{v}_i) = (\alpha|\mathbf{v}_i^{'}|^2 + \beta|\mathbf{v}_i^{''}|^2)/2$, where $\mathbf{v}_i^{'}$ and $\mathbf{v}_i^{''}$ are the first and second (discrete) derivatives of $\mathbf{v}_i$, given positive weights α and β respectively. The external energy E_{ext} at each snaxel is typically defined as $E_{\text{ext}}(\mathbf{v}_i) = -|\nabla(G_\sigma * I(x_i, y_i))|^2$,where G_σ is a Gaussian kernel with standard deviation σ, and ∇ stands for the gradient operator. $G_\sigma * I(x_i, y_i)$ is the convolves the image with a Gaussian filter to smooth it. These energies produce forces on the snake, causing its contour to move and change shape to minimize the energy E_{snake}.

In practice, it is not easy to use such a definition for the external energy in natural cluttered video sequences. Convergence of the contour to the correct solution can easily be affected wherever parts of the target and background have similar appearance, or wherever image contrast is weak. To help correctly locate the contour, we add a further term, the gradient of the confidence map, into the external energy. This is now defined as:

$$E_{\text{ext}}(\mathbf{v}_i) = -\gamma|\nabla(G_\sigma * I(x_i, y_i))|^2 - \lambda|\nabla(G_{\sigma'} * I_{\text{conf}}(x_i, y_i))|^2 \tag{2}$$

where I_{conf} is the confidence map produced by the IERF classifiers (see Section 2), and γ, λ are weighting factors that control the effect of each energy term. Because the confidence map is less noisy than the original video frame, the Gaussian filters have different deviations, and $\sigma^{'} < \sigma$.

In summary, weights α and β control the relative importance of the elasticity and rigidity of the curve, while the γ and λ control attraction of the snake towards the target boundary under as determined by gradients of the video image, and by the gradient of the confidence map. Each weight is in $[0, 1]$.

Since the confidence map provides a much more accurate location for the target than the video pixels, the new external energy more successfully attracts the snake to the right place, as we show in our experiments later.

4 Multi-cue fusion mixed cascade particle filter

To track the object contour, we adopt a Bayesian inference framework to estimate each target state sequentially. Below we first review the standard particle filter, and then describe our new multi-cue mixed cascade particle filter tracking framework.

Particle filtering is a sequential importance sampling algorithm for estimating properties of hidden variables in a hidden Markov model, given observations.

Given some set of observations of feature values $\mathbf{O}_t = (\mathbf{o}_1, \cdots, \mathbf{o}_t)$ for a target up to time t, the aim of a particle filter system is to estimate the posterior $p(\mathbf{x}_t|\mathbf{O}_t)$, where $\mathbf{x}_t$ is the state of the target at time t, based on the *observation* model (the *likelihood*) $p(\mathbf{o}_t|\mathbf{x}_t)$ and the *dynamic* model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$:

$$p(\mathbf{x}_t|\mathbf{O}_t) \propto p(\mathbf{o}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{O}_{t-1}) dx_{t-1} \tag{3}$$

The tracking result is obtained as the *maximum a-posteriori* (MAP) estimate, which is: $\mathbf{x}_t^* = \arg\max p(\mathbf{x}_t|\mathbf{O}_t)$.

The particle filter approach approximates the integral in Equ. 3 by using a set of weighted samples (particles) $\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N$, where each $\mathbf{x}_t^i$ is an estimate of state and w_t^i is the corresponding weight. These particles are generated during the initialization stage, and evolve continually.

In our tracking system, the object state is represented as

$$\mathbf{x}_t = < \mathbf{c}_t, \mathbf{c}_{t-1}, s_t, s_{t-1}, \mathbf{S}_t, \mathbf{S}_{t-1} >,$$

where $\mathbf{c}_t$, s_t, and $\mathbf{S}_t$, are the coordinates of the centroid of the target region (defined as a rectangle here), the size of the target region, and the set of snaxels at time t. The target region rectangle is initialised by drawing it on the first frame.

4.1 Multiple observation models

Combining multiple cues can significantly improve tracking performance, as different cues can complement each other and thus overcome failures of individual cues. Therefore, we use multiple observation models which represent both the object's appearance and and contour.

We assume the likelihood probability to be

$$p(\mathbf{o}_t|\mathbf{x}_t) = \prod_{i=1}^{3} p(\mathbf{o}_{t,i}|\mathbf{x}_t) = p_{\text{classifier}}(\mathbf{o}_{t,1}|\mathbf{x}_t) p_{\text{intra\_deform}}(\mathbf{o}_{t,2}|\mathbf{x}_t) p_{\text{inter\_deform}}(\mathbf{o}_{t,3}|\mathbf{x}_t) \tag{4}$$

where $\mathbf{o}_t = \{\mathbf{o}_{t,1}, \mathbf{o}_{t,2}, \mathbf{o}_{t,3}\}$ represents the state at time t, including the appearance of the object, the intra-frame deformation and the inter-frame deformation separately, and $p_{\text{classifier}}$, $p_{\text{intra\_deform}}$, $p_{\text{inter\_deform}}$ are the corresponding likelihoods.

Given the output H_F of the IERF classifiers described in Section 2, the observation likelihood $p_{\text{classifier}}$ is defined as:

$$p_{\text{classifier}}(\mathbf{o}_{t,1}|\mathbf{x}_t) = \frac{1}{1 + \exp(-\tau H_F(\mathbf{x}_t))}, \tag{5}$$

where τ is a control parameter.

Deformation of the object contour can be considered in two ways, within the current frame, and between consecutive frames, for which we use separate observation models. The likelihood of intra-frame deformation is defined according to the snake energy E_{snake} given by Equ. 1:

$$p_{\text{intra\_deform}} \propto \exp(-E_{\text{snake}}/\sigma_s^2), \tag{6}$$

where σ_s is a user defined weighting factor.

The likelihood of inter-frame deformation is calculated using the bending energy of a thin plate spline (TPS) model.

$$p_{\text{inter\_deform}} \propto \exp(-E_{\text{bending}}/\sigma_b^2), \tag{7}$$

where σ_b is a user specified weight.

To find E_{bending}, first, we find correspondences ψ between the sets of snaxels for contours in consecutive frames, using a multiple order graph matching method [1]. A multiple order approach leads to more consistent relationships with more accurate and robust results than using a single order. On the other hand, multiple order graph matching requires fewer iterations than bipartite graph matching.

We solve a linear equation to find the TPS coefficients. Let the two sets of snaxels be $\mathbf{S}_{t-1} = \{s_{t-1}^i\}_{i=1}^{N_{t-1}}$ and $\mathbf{S}_t = \{s_t^i\}_{i=1}^{N_t}$ in frames at times $t-1$ and t, with N_{t-1} and N_t snaxels respectively. We use two independent interpolation functions f_{m_x} and f_{m_y} to model the coordinate transformations between them. These functions satisfy $x_t^{\psi(i)} = f_{m\_x}(x_{t-1}^i, y_{t-1}^i)$ and $y_t^{\psi(i)} = f_{m\_y}(x_{t-1}^i, y_{t-1}^i)$, where $(x_{t-1}^i, y_{t-1}^i) \in \mathbf{S}_{t-1}$ is the image coordinate of the snaxel s_{t-1}^i at time $t-1$, while $(x_t^{\psi(i)}, y_t^{\psi(i)})$ is the image coordinate of snaxel $s_t^{\psi(i)}$ which is the corresponding point to s_{t-1}^i. Because $E_{\text{bending}} \approx \mathbf{b}^T K \mathbf{b}$ [9], and letting $K_{ij} = U(\|s_{t-1}^i - s_{t-1}^j\|)$, where $U(r) = r^2 \log r^2$, the TPS coefficients $\mathbf{b}$ can be obtained by solving the following linear equation:

$$\begin{bmatrix} K & D \\ D & \mathbf{0}_{3\times 3} \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{0}_{3\times 2} \end{bmatrix} \tag{8}$$

where the i^{th} row of D is $(1, x_{t-1}^i, y_{t-1}^i)$, and the i^{th} row of $\mathbf{v}$ is $(x_t^{\psi(i)}, y_t^{\psi(i)})$, from the interpolation conditions $(f_{m\_x}(x_{t-1}^i, y_{t-1}^i), f_{m\_y}(x_{t-1}^i, y_{t-1}^i)) = v_i$. The first and second columns of $\mathbf{b}$ satisfy

$$\sum_{i=1}^{N_{t-1}} b(i,1) = \sum_{i=1}^{N_{t-1}} b(i,2) = 0 \tag{9}$$

$$\sum_{i=1}^{N_{t-1}} b(i,1) x_{t-1}^i = \sum_{i=1}^{N_{t-1}} b(i,1) y_{t-1}^i = 0 \tag{10}$$

$$\sum_{i=1}^{N_{t-1}} b(i,2) x_{t-1}^i = \sum_{i=1}^{N_{t-1}} b(i,2) y_{t-1}^i = 0 \tag{11}$$

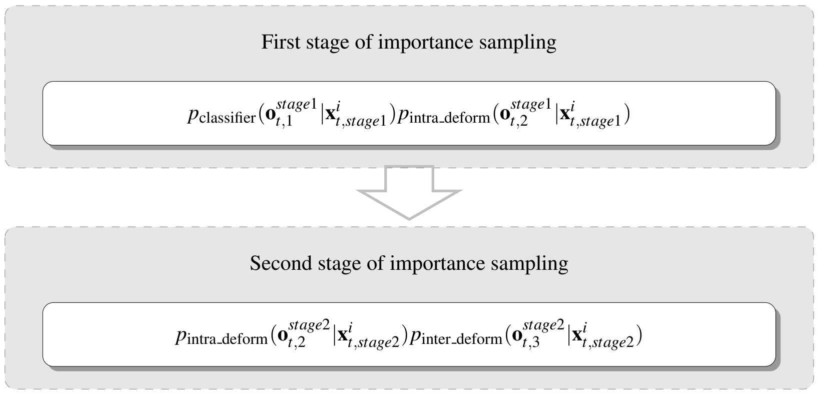

First stage of importance sampling
$p_{\text{classifier}}(\mathbf{o}_{t,1}^{stage1}|\mathbf{x}_{t,stage1}^{i})p_{\text{intra\_deform}}(\mathbf{o}_{t,2}^{stage1}|\mathbf{x}_{t,stage1}^{i})$
Second stage of importance sampling
$p_{\text{intra\_deform}}(\mathbf{o}_{t,2}^{stage2}|\mathbf{x}_{t,stage2}^{i})p_{\text{inter\_deform}}(\mathbf{o}_{t,3}^{stage2}|\mathbf{x}_{t,stage2}^{i})$

Fig. 2 Mixed cascade importance sampling, showing the two different stages of the importance sampling process.

4.2 Two-stage mixed cascade importance sampling

Having introduced multiple individual cues above, a standard particle filter can be directly adopted by updating every particle weight using the observation model: $p(\mathbf{o}_t|\mathbf{x}_t) = \prod_{i=1}^{3} p(\mathbf{o}_{t,i}|\mathbf{x}_t)$. Instead of updating particle weights directly according to the above observation model, we use two stages of importance sampling to fuse all the observations efficiently in a similar way to the cascade method used in [12], but we do not do so simply in sequence.

As shown in Fig. 2, in the first stage, the particle weights are calculated by using the appearance model and the intra-frame deformation model. We combine the particle weights obtained from the appearance model, and those from the intra-frame deformation model, $p_{\text{classifier}} \cdot p_{\text{intra\_deform}}$, allowing us to preserve the particles which have contours close to the real location. Then in the second stage, we update the particle weights again using the intra-frame deformation model and inter-frame deformation models, because the inter-frame deformation model $p_{\text{inter\_deform}}$ will be meaningful when two contours in two particles are not quite different.

Our method avoids much unnecssary computation, compared to the obvious method of computing all three likelihoods for every particle at each step of updating the particle weight. Because our method does not fuse all observation models fully in sequence like the method in [12], we call it *mixed cascade* importance sampling. See Algorithm 1.

The output $\{\mathbf{x}_{t,stage2}^{i}, w_{t,stage2}^{i}\}_{i=1}^{N_2}$ of Algorithm 1 is used for state estimation.

4.3 Improved dynamic model

An auto-regressive (AR) process [3], like the first order AR model in Equ. 12, is usually used as the dynamic model in particle filter algorithms. A is the state transition matrix, and B defines the transfer radius of the particle. $\mathbf{r}_{t-1}$ represents multivariate

Algorithm 1 Mixed cascade importance sampling (IS)

```
 1: Calculate the weight $w^i_{t,stage1}$ of each particle in the first stage of importance sampling:
 2:     for $i = 1 \cdots N_1$
 3:         $w^i_{t,stage1} = p_{\text{classifier}}(\mathbf{o}^{stage1}_{t,1}|\mathbf{x}^i_{t,stage1})p_{\text{intra\_deform}}(\mathbf{o}^{stage1}_{t,2}|\mathbf{x}^i_{t,stage1})$
 4:     endfor
 5: Resample according to the weights $\mathbf{w}_{t,stage1} = \{w^i_{t,stage1}\}^{N_1}_{i=1}$:
 6:     Generate $l_j$ from $\{w^i_{t,stage1}\}^{N_1}_{i=1}$, and replace $\{\mathbf{x}^i_{t,stage1}, w^i_{t,stage1}\}^{N_1}_{i=1}$ by $\{\mathbf{x}^{l_i}_{t,stage2}, 1/N_2\}^{N_2}_{j=1}$
 7: Disturb the states of all particles in the first stage:
 8:     for $i = 1 \cdots N_2$
 9:         Get $\mathbf{x}^i_{t,stage2}$ from $g(\mathbf{x}_{t,stage2}|\mathbf{x}^i_{t,stage1})$
10:         Let $\lambda^i = g(\mathbf{x}^i_{t,stage2}|\mathbf{x}^i_{t,stage1})$, where $g$ is a 0-mean Gaussian function
11:     endfor
12: Calculate the weight $w^i_{t,stage2}$ of each particle in the second stage of the importance sampling process:
13:     for $i = 1 \cdots N_2$
14:         $w^i_{t,stage2} = p_{\text{intra\_deform}}(\mathbf{o}^{stage2}_{t,2}|\mathbf{x}^i_{t,stage2})p_{\text{inter\_deform}}(\mathbf{o}^{stage2}_{t,3}|\mathbf{x}^i_{t,stage2})$
15:     endfor
16: Normalize $\sum^{N_2}_{i=1} w^i_{t,stage2} = 1$.
    Note: $\mathbf{o}_t = \{\mathbf{o}^{stage1}_{t,1}, \mathbf{o}^{stage1}_{t,2}, \mathbf{o}^{stage2}_{t,2}, \mathbf{o}^{stage2}_{t,3}\}$ and $\mathbf{x}_t = \{\mathbf{x}_{t,stage1}, \mathbf{x}_{t,stage2}\}$ are the observations and states
    at time $t$. $\mathbf{o}_{t,1}$ is the appearance model. $\mathbf{o}_{t,2}$ and $\mathbf{o}_{t,3}$ are the intra-frame deformation model and the
    inter-frame deformation model.
```

Gaussian random noise.

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + B\mathbf{r}_{t-1}. \tag{12}$$

The main drawback of such a model is that it does not take into account the most recent observation, and a constant velocity model is usually not suitable for real applications. Here we use information obtained from optical flow to revise Equ. 12.

As shown in Fig. 3, optical flow represents pixel motion in a video sequence. In recent years, optical flow computation has become more and more accurate, and GPU computation has provided a number of real-time optical flow algorithms [13, 14]. The optical flow in a frame at time t gives a pixel motion vector: $\mathbf{v}_{\text{img}} = \{v_x, v_y\}$, where v_x, v_y are optical flows along the x and y axes. An improved first order AR model is given by:

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + \delta\mathbf{x}_t + B\mathbf{r}_{t-1}, \tag{13}$$

where $\delta\mathbf{x}_t$ is the update obtained from $\mathbf{v}_{\text{img}}$. The centroid of the target region $\mathbf{c}_t$ satisfies

$$\mathbf{c}_t = A\mathbf{c}_{t-1} + \bar{\mathbf{v}}_{\text{img}}(\mathbf{c}) + B\mathbf{r}_{t-1} \tag{14}$$

where $\bar{\mathbf{v}}_{\text{img}}(c_t) = \{\bar{v}_x(c_t), \bar{v}_y(c_t)\}$ is the average optical flow around point $\mathbf{c}_t$. Each snaxel $\mathbf{d} \in \mathbf{S}_t$ also satisfies

$$\mathbf{d}_t = A\mathbf{d}_{t-1} + \bar{\mathbf{v}}_{\text{img}}(\mathbf{d}_t) + B\mathbf{r}_{t-1} \tag{15}$$

where $\bar{\mathbf{v}}_{\text{img}}(\mathbf{d}_t) = \{\bar{v}_x(\mathbf{d}_t), \bar{v}_y(\mathbf{d}_t)\}$ is the average optical flow around point $\mathbf{d}_t$.

Our overall multi-cue based discriminative contour tracking method is shown in Algorithm 2.

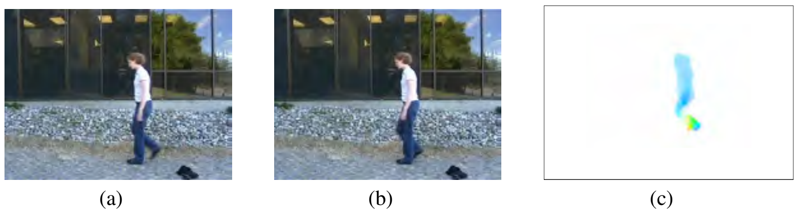


Fig. 3 Optical flow field for two consecutive frames. Image (a) and (b) are two consecutive frames. The color in (c) displays the angle and brightness while white denotes no motion.

Algorithm 2 Multi-cue based discriminative contour tracking

```
1: At $t = 0$, label the initial target region $R_0$, extract the initial contour point set $\mathbf{S}_0$, initialize the particles $\{\mathbf{x}_0^i, w_0^i\}_{i=1}^{N_1}$.
2: for $t > 0$
3:     Detect the target by IERF classifiers, and get the confidence region $R_t$.
4:     Resample according to the weights of the particles.
5:     if $t = 1$
6:       Generate $l_j$ from $\{w_{t-1}^i\}_{i=1}^{N_1}$;
7:       Then replace $\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^{N_1}$ by $\{\mathbf{x}_{t-1}^{l_j}, 1/N_1\}_{j=1}^{N_1}$
8:     else
9:       Generate $l_j$ from $\{w_{t-1}^i\}_{i=1}^{N_2}$;
10:      Then replace $\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^{N_2}$ by $\{\mathbf{x}_{t-1}^{l_j}, 1/N_1\}_{j=1}^{N_1}$
11:    end if
12:    Calculate the optical flow $\mathbf{v}_{\text{img}} = \{v_x, v_y\}$ between frames at time $t$ and $t-1$.
13:    Predict the new states of the particles using the dynamic model in Eqns.14–15:
14:    Generate $\mathbf{x}_t^i$ from $p(\mathbf{x}_t|\mathbf{x}_{t-1} = \mathbf{x}_{t-1}^{l_j})$ and get $\{\mathbf{x}_t^i, 1/N_1\}_{i=1}^{N_1}$
15:    Calculate the weights $w_t^i$ of the particles using Algorithm 1
16:    Estimation the state of the target as $\hat{\mathbf{x}}_t = \sum_{i=1}^{N_2} w_t^i \mathbf{x}_t^i$
17:    Update the IERF classifiers.
18: end for
   Note: $N_1$ and $N_2$ are the total numbers of particles in two importance sampling stages.
```

We use the first frame to initialise all particles. A bounding rectangle is drawn around the target by the user. The region including the target plus the contour is regarded as a particle. All particles in the first frame share the same region parameters and same contour. The region outside the target rectangles is marked as background.

5 Experimental results

In this section, we first demonstrate the performance of our discriminative parametric snake model and then the improved dynamic model. Finally, we verify the whole multi-cue based discriminative contour tracking method.

For all experiments, we used the same configuration of IERF classifiers. For classification, each video frame is divided into 9×9 patches. An HSL color histogram with 8 bins per patch is used to represent color features and a HoG histogram with 9 orientation bins per patch is used to represent the HoG features. 30 trees are used for each feature space. Our least-recently-used strategy is to remove samples more than T frames old from the forest. Here we choose T from 10 to 50 depending on object appearance and background changes. The GPU implementation of the Huber-L1 method in [13] was used for optical flow calculation.

For all experiments, The model parameters were set to $\tau = 1.0$, $\sigma_s = 0.85$ and $\sigma_b = 0.08$. The deviation σ of the Gaussian filter used for the original video frame was set to 0.5, while the deviation $\sigma^{'}$ for the confidence map was set to 0.3.

5.1 Test of discriminative parametric snake model

To test the performance of our discriminative parametric snake model, we implemented two different contour tracking methods. Method 1 shares the same tracking framework proposed in this paper but using the usual parametric snake model (using image gradients as the external energy). Method 2 is our proposed multi-cue fusion mixed cascade particle filter tracking method with the proposed snake model.

The weights used for the snake model in method 1 were $\alpha = 0.4$, $\beta = 0.2$, $\gamma = 0.6$; the weights in method 2 were $\alpha = 0.4$, $\beta = 0.2$, $\gamma = 0.6$, $\lambda = 0.8$. Both methods shared the same particle filter tracking framework, with 20 particles used in each importance sampling stage.

Fig. 4 shows tracking results on a pedestrian sequence. The background has a strong texture which seriously affects tracking methods. Method 1, using the usual snake model, failed on frame 12, with a small region of cluttered background inside the contour on the person's leg. In subsequent frames 14 16, the contours did not converge correctly, and the part in error got larger. However, our method avoided the distraction provided by the cluttered background. The discriminative model helps the extracted contours closely follow the real edge of the pedestrian.

Fig. 5 shows tracking results on another pedestrian sequence. Around frame 254 the target person occludes another person moving across. In method 1, the contour is disturbed by the strong vertical edge of the occluded person. Again, in our method, the contours are quite accurate and the curves enclose only the target person: the improved external energy term forces the contour to converge towards the real edge of the target.

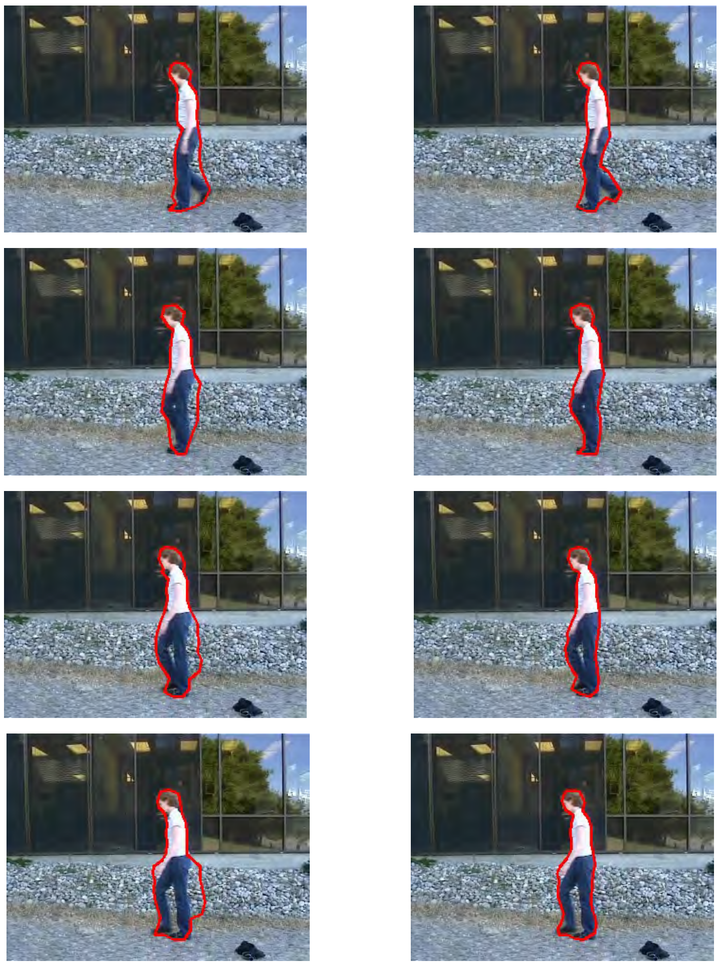
These and other experiments have verified the stabilizing effect of our discriminative parametric snake model.

5.2 Test of improved dynamic model

We next tested the performance of the improved dynamic model used in the particle filter framework. We implemented two different contour tracking methods. Method 1 uses the same tracking framework but using the ordinary AR model. Method 2 was our proposed tracking method with the improved dynamic model.

Both methods used the same particle filter tracking framework, with 20 particles used in each importance sampling stage. The weights of the snake model were $\alpha = 0.4$, $\beta = 0.2$, $\gamma = 0.6$, $\lambda = 0.8$ for each model.

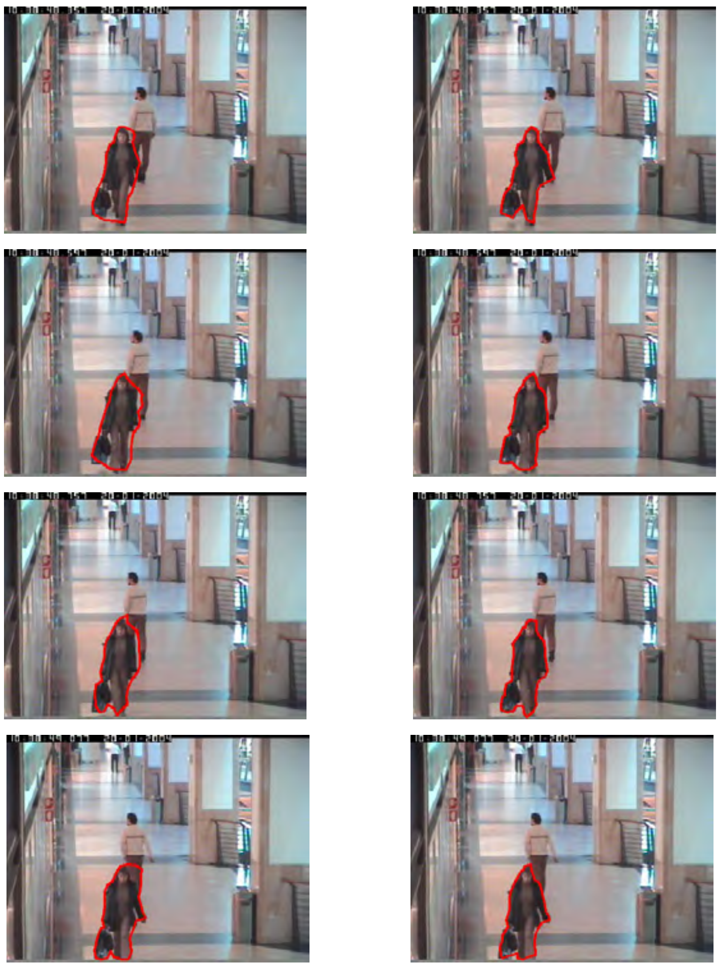
The targets in the video sequences used in this section all moved quickly. Fig. 6 contains a girl's head with a fast motion and Fig. 7 contains a running person. It is



(a) Result of method 1 (b) Result of method 2

Fig. 4 Pedestrian sequence with cluttered background. The red closed curve indicates the contour. From top to bottom are frames 12, 14, 15 and 16. Column (a) is the result of method 1, (b) is the result of method 2 (our method).

clear that method 1 failed to track these targets with fast or irregular motion. From frame 18 in Fig. 6, the face contour includes part of the background region and totally loses the target in the successive frames. In Fig. 7 the person changes location very fast and method 1 cannot find the right location just a few frames after the start. The improved dynamic model used in the tracking framework in method 2 can predict correct locations in successive frames, giving accurate tracking results as seen in Figs. 6 and 7.



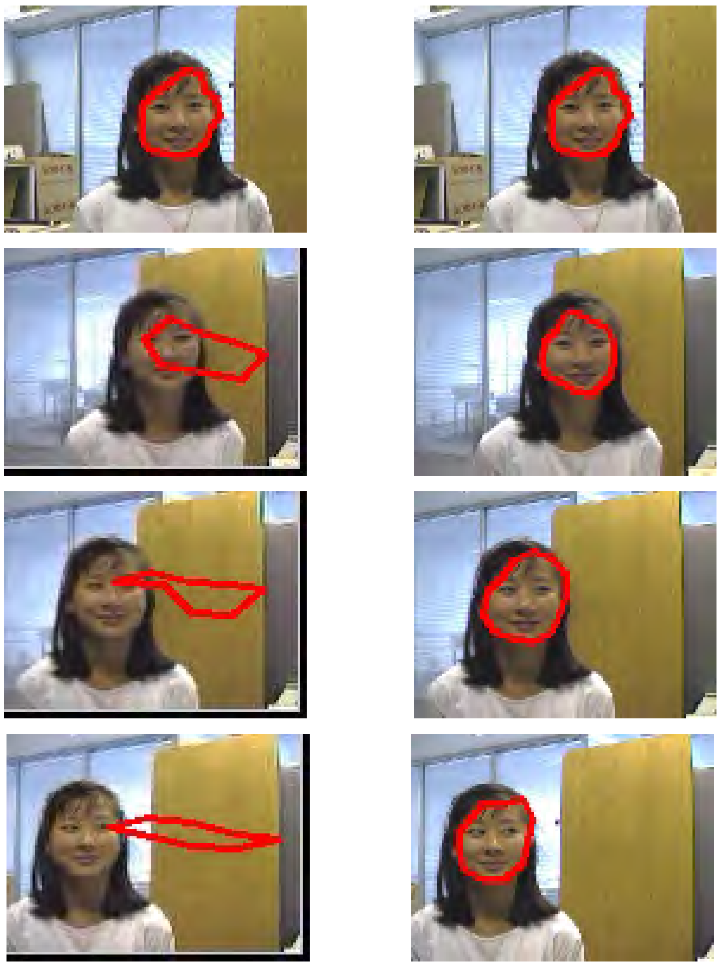
(a) Result of method 1 (b) Result of method 2

Fig. 5 Pedestrian sequence with occlusion. The red closed curve indicates the contour. From top to bottom are the frames 254, 260, 265 and 272. Column (a) is the result of method 1, (b) is the result of method 2 (our method).

5.3 Test on the proposed contour tracking method

This section demonstrates the performance of our proposed algorithm on several different challenging video sequences. For comparison, we also implemented the *Deform PF-MT* tracking method [15]. In both methods, the initial contour was always located manually in the first frame.

For our method, the weighting factors of the snake energy were again $\alpha = 0.4$, $\beta = 0.2$, $\gamma = 0.6$, $\lambda = 0.8$, and the total numbers of particles in two importance sampling stages were both set to 20.



(a) Result of method 1 (b) Result of method 2

Fig. 6 Head sequence with fast motion. The red closed curve indicates the contour. From top to bottom are the frames with No.0, 18, 22 and 25. Column (a) is the result of method 1, (b) is the result of method 2 (our method).

Fig. 8 shows the tracking result on a face sequence. The camera is moving, so the constant velocity assumption in [15] finds it difficult to predict the object state. Furthermore, the illumination changes significantly, so the observation model in [15] finds it hard to describe the appearance of the target. Thus the Deform PF-MT method soon failed, starting from frame 8. Because of the IERF classifiers, our method is adaptive to appearance changes in the target. Furthermore, the improved dynamic model predicts the state well, making our tracking method more robust to camera shake.

Fig. 9 shows the tracking result on another face sequence. The target in the video sequence is occluded by a magazine in many frames, which makes it hard for the the

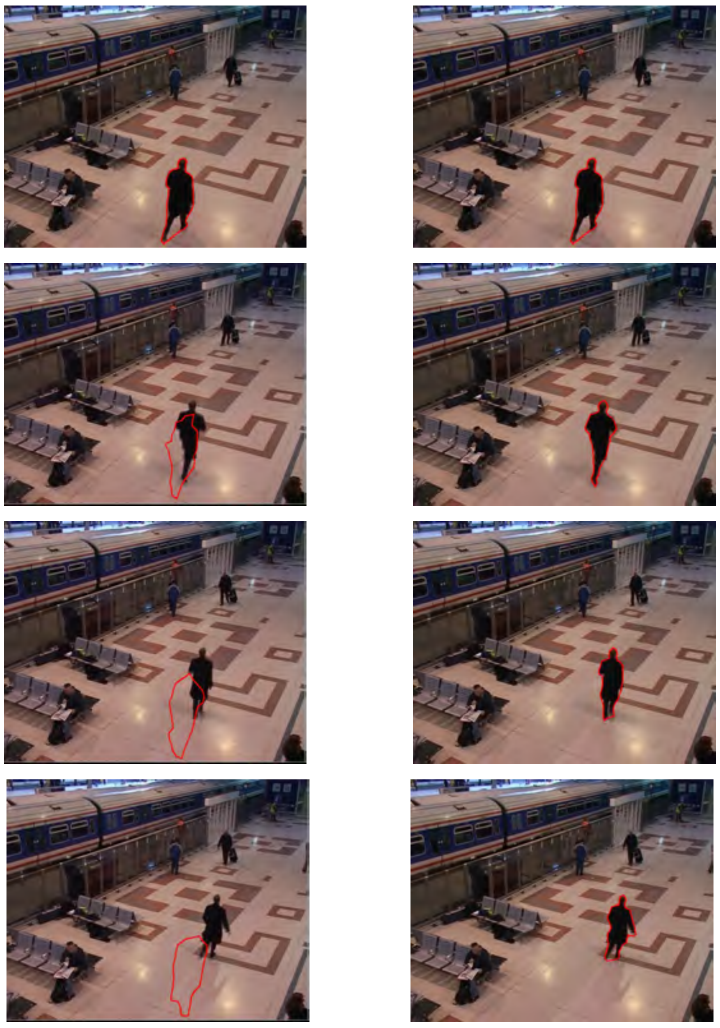

(a) Result of method 1 (b) Result of method 2

Fig. 7 Running pedestrian sequence. The red closed curve indicates the contour. From top to bottom are the frames with No.1200, 1206, 1213 and 1220. Column (a) is the result of method 1, (b) is the result of method 2 (our method).

Deform PF-MT method to find the correct region of the target. However, our method is robust to the occlusion as it uses an adaptive discriminative model.

Fig. 10 shows results on a vehicle sequence. The camera is again unfixed, and the car in the scene moves quickly, making it is difficult for the the Deform PF-MT method to predict the object state. Furthermore, illumination changes significantly (due to the bridge), making it hard for the the Deform PF-MT method to describe the appearance of the target. The Deform PF-MT method failed as the illumination changed, and the contour drifted to another region in the frame—see frame 234. Be-

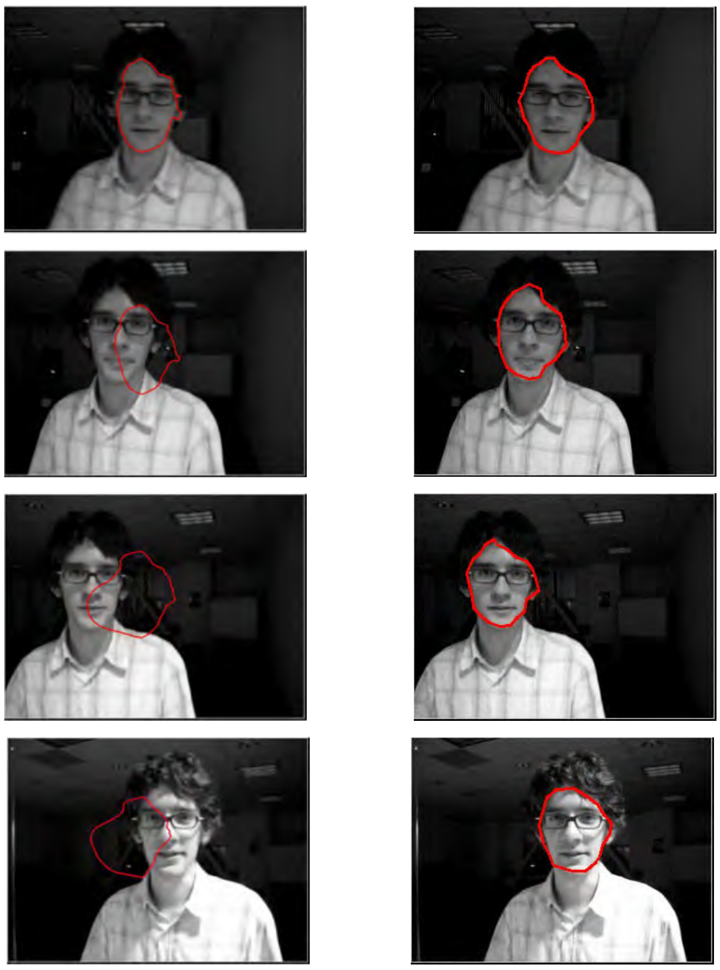
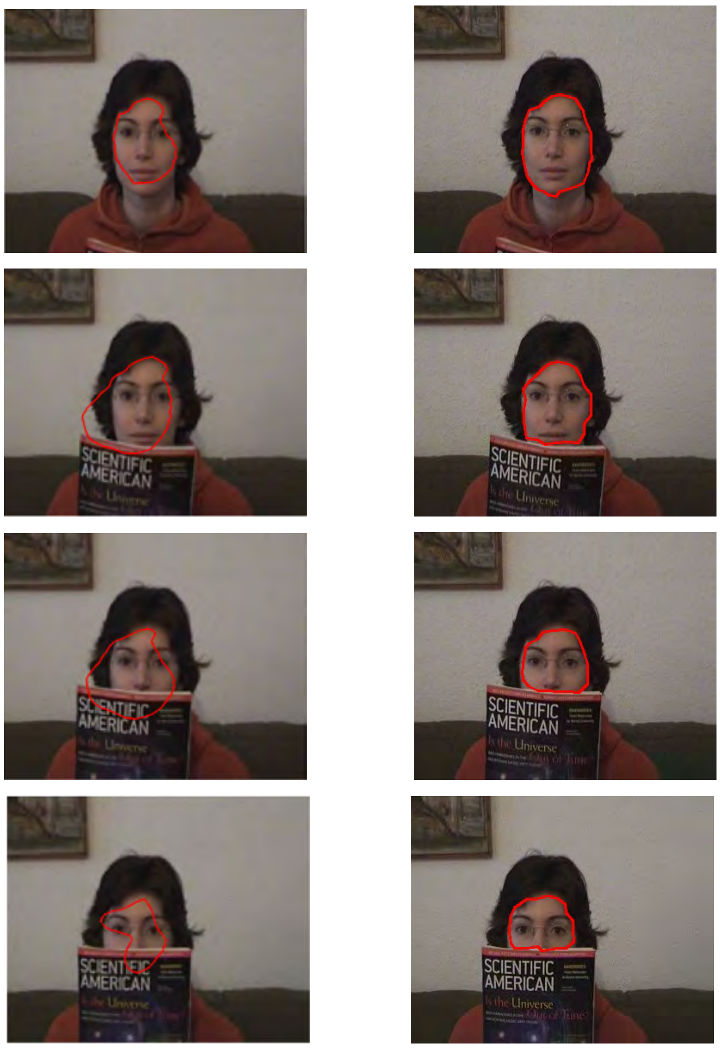

(a) Result of method [15] (b) Result of our method

Fig. 8 Face sequence with camera shake and varying illumination. The red closed curve indicates the contour. From top to bottom are the frames 1, 8, 18 and 38. Column (a) is the result of the the Deform PF-MT method, (b) is the result of our tracking method.

cause of the IERF model, our method is adaptive to the illumination change, and the improved dynamic model also predicts the state well, allowing our tracking method to track the fast moving car.

The number of frames successfully tracked for each these challenging sequences is given in Table 1, clearly demonstrating the advantage of our new approach.

Our method was implemented using un-optimized C++ code and all experiments were run on a 2.3GHz, 1GB PC. The average speed was 3 4 fps for colour sequences of 320×240 resolution. However the IERF classifier can be parallelized, and our method could thus run much faster.



(a) Result of method [15] (b) Result of our method

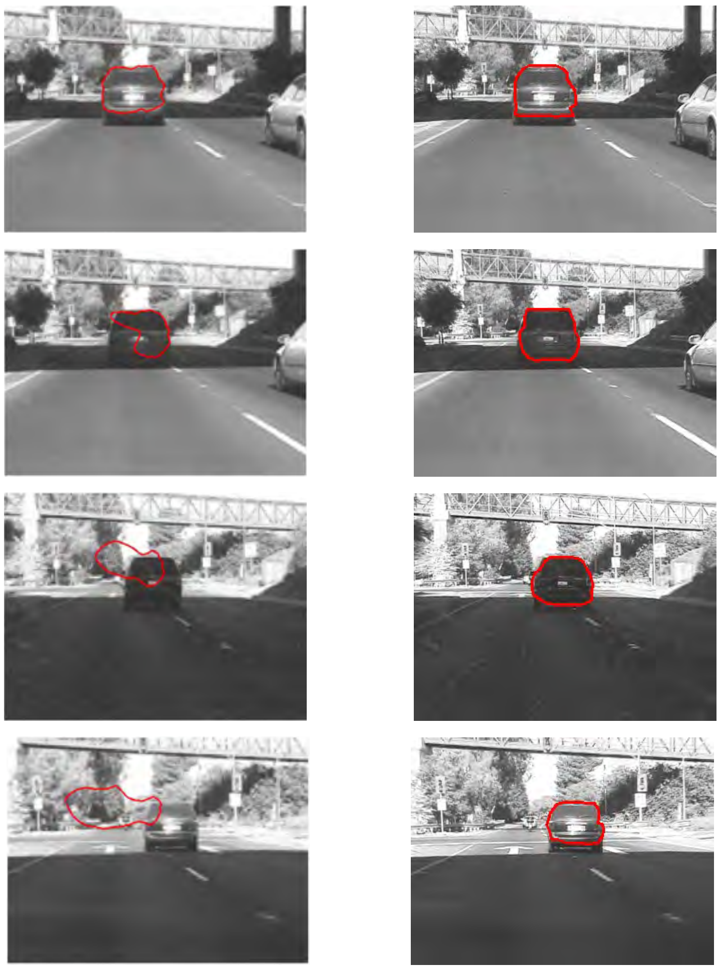
Fig. 9 Face sequence with occlusion. The red closed curve indicates the contour. From top to bottom are the frames 264, 272, 276 and 302. Column (a) is the result of method [15], (b) is the result of our tracking method.

Table 1 Number of successfully tracked frames for various sequences, and total number of frames.

	Our method	Method in [15]
Face sequence 1	50/50	5/50
Face sequence 2	62/62	6/62
Vehicle sequence	55/55	8/55

6 Conclusion

This paper has presented a multi-cue discriminative object contour tracking algorithm with two main contributions.



(a) Result of method [15] (b) Result of our method

Fig. 10 Vehicle sequence with camera shake and seriously varying illumination. The red closed curve indicates the contour. From top to bottom are the frames with No.180, 188, 210 and 234. The column (a) is the result of method [15], while the column (b) is the result of our tracking method.

The first is a mixed cascade particle filter tracking algorithm using multiple observation models to improve the accuracy and stability of object contour tracking. An incremental extremely random forest classifier (IERF) is used to provide an incremental model of target appearance. To describe the deformation of the object contour, we use both inter-frame and intra-frame deformation models. The former is based on thin plate spline bending energy, using multiple order graph matching to determine correspondence between contours in consecutive frames. The latter is given by the contour evolution energy in the current frame. To fuse these observations efficiently, a mixed cascade importance sampling process is used. Optical flow is used when updating the dynamic model in the particle filter algorithm to achieve further improvements.

The second contribution is a novel contour evolution energy. We integrate an incremental learning discriminative model into the parametric snake model to improve the performance of the contour evolution process.

Results on challenging video sequences demonstrate the effectiveness and robustness of our method.

References

1. Aiping, W., Sikun, L., Liang, Z.: Multiple order graph matching. In: Proceedings of the 10th Asian conference on Computer vision (ACCV) - Volume Part III, p. 471C482 (2002)
2. Blake, A., Isard, M.: Active Contours. Springer Verlag (1998)
3. Brockwell, P., Davis, R.: Time Series: Theory and Methods (Springer Series in Statistics). Springer (2009)
4. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. International Journal of Computer Vision **22**(1), 61–79 (1997)
5. Chen, Q., Sun, Q.S., Heng, P.A., Xia, D.S.: Two-stage object tracking method based on kernel and active contour. IEEE Transactions on Circuits and Systems for Video Technology **20**, 605–609 (2010)
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 886–893 (2005)
7. Dambreville, S., Rathi, Y., Tannenbaum, A.: Tracking deformable objects with unscented kalman filtering and geometric active contours. In: American Control Conference, pp. 2856–2861 (2006)
8. Frank, Y.S., Kai, Z.: Locating object contours in complex background using improved snakes. Comput. Vis. Image Underst. **105**(2), 93–98 (2007)
9. Gianluca, D., Serge, B.: Approximate thin plate spline mappings. In: Proceedings of the 7th European Conference on Computer Vision (ECCV) -Part III, pp. 21–31 (2002)
10. Isard, M., Blake, A.: Condensation – conditional density propagation for visual tracking. Journal of the Society for Industrial and Applied Mathematics (1998)
11. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision **1**(4), 321–331 (1988)
12. Li, Y., Ai, H., Yamashita, T., Lao, S., Kawade, M.: Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. IEEE Trans. Pattern Anal. Mach. Intell. **30**, 1728–1740 (2008)
13. Manuel, W., Werner, T., Thomas, P., Wedel, A., Cremers, D., Horst, B.: Anisotropic huber-l1 optical flow. In: Proceedings of British Machine Vision Conference (BMVC), pp. 1–11 (2009)
14. Marzat, J., Dumortier, Y., Ducrot, A.: Real-time dense and accurate parallel optical flow using cuda. In: Proceedings of The 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG2009, pp. 105–111 (2009)
15. Namrata, V., Yogesh, R., Anthony, Y., Allen, T.: Deform pf-mt: particle filter with mode tracker for tracking nonaffine contour deformations. IEEE Trans. Image Processing **19**(4), 841–857 (2010)
16. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer (2003)
17. Peterfreund, N.: Robust tracking of position and velocity with kalman snakes. IEEE Trans. Pattern Anal. Mach. Intell. **21**(6), 564–569 (1999)
18. Rathi, Y., Vaswani, N., Tannenbaum, A., Yezzi, A.: Tracking deforming objects using particle filtering for geometric active contours. IEEE Trans. Pattern Anal. Mach. Intell. **29**, 1470–1475 (2007)
19. Sethian, J.A.: Level Set Methods and Fast Marching Methods. Cambridge University Press (1999)
20. Terzopoulos, D., Szeliski, R.: Tracking with kalman snakes. In: Active Vision, pp. 3–20. MIT Press (1992)
21. Wang, A., Wan, G., Cheng, Z., Li, S.: An incremental extremely random forest classifier for online learning and tracking. In: IEEE International Conference on Image Processing (ICIP), pp. 1449 – 1452 (2009)
22. Wei, L., Xiaoqin, Z., Jun, G., Weiming, H., Haibin, L., Xue, Z.: Discriminative level set for contour tracking. In: Proceedings of the International Conference on Pattern Recognition (ICPR) (2010)
23. Xu, C., Prince, J.: Snakes, shapes and gradient vector flow. IEEE Transactions on Image Processing **7**(3), 359–369 (1998)