Tracking in Low Frame Rate Video: A Cascade Particle Filter with Discriminative Observers of Different Lifespans

Yuan Li, Haizhou Ai
Computer Science and Technology Dept.
Tsinghua University, Beijing, China
ahz@mail.tsinghua.edu.cn

Takayoshi Yamashita, Shihong Lao, Masato Kawade
Sensing and Control Technology Laboratory
Omron Corporation, Kyoto, Japan
{takayosi|lao|kawade}@ari.ncl.omron.co.jp

Abstract

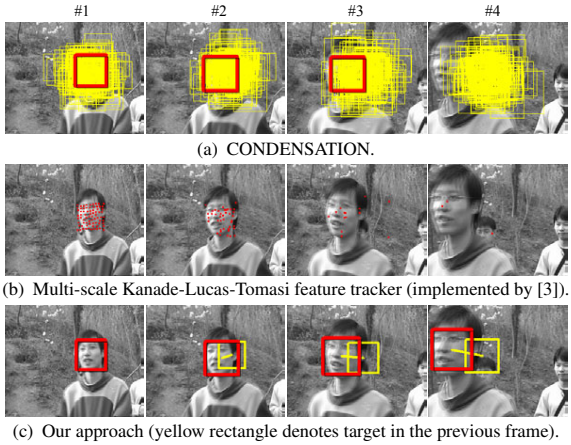
Tracking object in low frame rate video or with abrupt motion poses two main difficulties which conventional tracking methods can barely handle: 1) poor motion continuity and increased search space; 2) fast appearance variation of target and more background clutter due to increased search space. In this paper, we address the problem from a view which integrates conventional tracking and detection, and present a temporal probabilistic combination of discriminative observers of different lifespans. Each observer is learned from different ranges of samples, with different subsets of features, to achieve varying level of discriminative power at varying cost. An efficient fusion and temporal inference is then done by a cascade particle filter which consists of multiple stages of importance sampling. Experiments show significantly improved accuracy of the proposed approach in comparison with existing tracking methods, under the condition of low frame rate data and abrupt motion of both target and camera.

1. Introduction

Tracking in low frame rate (LFR) video is a practical requirement of many of today's real-time applications such as in micro embedded systems and visual surveillance. The reason is various: hardware costs, LFR data source, online processing speed which upper-bounds the frame rate, etc. Moreover, for a tracking system, LFR condition is equivalent to abrupt motion, which is often encountered but hard to cope with.

Although the body of literature regarding tracking is huge, most existing approaches (except a few categories) cannot be readily applied to LFR tracking problems, either because of the slow speed or the vulnerability to motion and appearance discontinuity caused by LFR data.

The key notion of our solution is that detection and tracking can be integrated to overcome this difficulty. As two extremes, conventional tracking facilitates itself with every possible assumption of temporal continuity, while detection aims at the universal description or discrimination of the target from the others. In LFR tracking, the continuity of target is often too weak for conventional tracking (Figure 1); meanwhile, applying reliable detection over a large search space is often unaffordable, neither is it capable of identifying target through frames due to neglect of context.



(a) CONDENSATION.

(b) Multi-scale Kanade-Lucas-Tomasi feature tracker (implemented by [3]).

(c) Our approach (yellow rectangle denotes target in the previous frame).

Figure 1. Tracking 4 consecutive frames in a 5pfs video.

Therefore it is desirable to have the two complementing each other – achieving strong discriminative power yet maintaining the grasp of weak spatial-temporal continuity. We approach this by adopting a series of observation models (observers) with different "lifespan"s. By "lifespan" we mean the learning period and service period of an observer. *E.g.*, a two-frame template matching tracker can be viewed as an observer with learning period and service period both of one frame; and an offline trained detector can be view as one with a very long lifespan, in the sense that it is trained by as many exemplars as possible and is expected to apply to most sorts of appearance that belong to the target category (*e.g.*, face). The advantage lies in that observers with shorter lifespan can grab target appearance more specifically and rule out non-target candidates faster, the training cost is also lower since there is not much knowledge to mas-

1-4244-1180-7/07/$25.00 ©2007 IEEE

ter; while those with longer lifespan can produce more accurate result and prevent the "drift" problem which is typical of an online learning system.

Further in organizing these observers, rather than adopting a cascade (which is a common approach in detection literature, however problematic in our approach as will be shown) and a standard particle filter (which is also one of the most celebrated tracking frameworks), we instead propose a "cascade particle filter" in the hope of combining the two successful ideas in both fields to satisfy the special needs of our particular problem.

In the next section, related work is briefly summarized. Section 3 is devoted to the learning of different observers. Section 4 first introduces the common approach of particle filter and reveals its deficiency in the view of LFR issue, and then describes the cascade particle filter and compares it with existing methods. After that are the experiment and conclusion sections.

2. Related work

LFR is in most cases equivalent to abrupt motion. However, a large part of traditional tracking approaches heavily depend on motion continuity. Particle filters [10] uses a dynamic model to guide the particle propagation within a limited sub-space of target state. Other methods based on iterative optimization such as mean shift [4] and Kanade-Lucas-Tomasi feature tracker [17] generally require the kernels or feature patches in consecutive frames to overlap with or be in a very close vicinity of each other. These, however, are presumptions too expensive under conditions of LFR or abrupt motion.

Several existing publications have been aware of this pitfall, may the motive be LFR tracking or not, the remedy has been quite unanimous: detection. Okuma *et al*. [14] uses a boosted detector to amend the trial distribution of particle filter. Such mixture trial distribution can also be found in many other works such as [12], although not aimed at LFR videos. Porilkli and Tuzel [15] extend the standard mean shift technique by optimizing around multiple kernels at motion areas detected by background modeling, to track in 1-fps camera-fixed surveillance video. These ideas can be concluded as using an independent detector to guide the search of an existing tracker when target motion becomes unpredictable.

Another extreme is to "detect and connect" [7][11]. Such approaches are of potential to deal with LFR tracking, because they first detect the objects of interest (sometimes track them through short sequences), and then construct trajectories (or connect trajectory fragments) by analysis of motion continuity, appearance similarity, etc. Real-time algorithms of this category are mostly limited in background-fixed scenes where a fast change detector is readily at hand.

The two categories above has a common drawback that they need a detector fast enough to be applied to large search areas (in most cases, the whole frame), partly because the detector is only loosely coupled with the tracker.

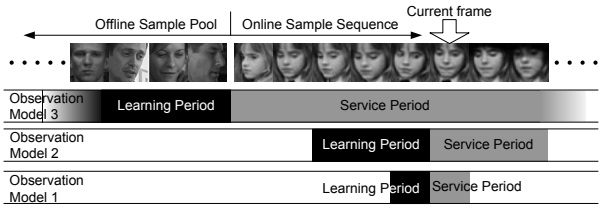


Figure 2. Life-spans of observation models (observers).

The seemingly most similar work to ours may be the multi-scale approaches for abrupt motion [8][3] and layered sampling of multi-scale likelihoods [16]. However, multi-scale approaches adopt essentially the same observation model on several down-scaled images, while our approach applies a pipeline of complementary observation models on the same image space. The latter's advantage lies in that discriminative power is increased and no risk of losing image information in down-scaling is induced.

Recently there has been a trend of introducing learning techniques into tracking problems, and tracking is viewed as a classification problem in the sense of distinguishing tracking target from the background. Representative publications include [2] [19], which have shown increased discriminative power of the tracker. Although none of them have targeted at LFR tracking, we will also incorporate online learning in our approach, only that online classifiers will be unified with offline ones to achieve enhanced robustness.

3. Learning discriminative observers

3.1. Representation

Following the convention of sequential Bayesian estimation, denote the state of the target object and the observation at time t by $\mathbf{x}_t$ and $\mathbf{z}_t$ respectively (in this section we suppress the subscript t when there is no ambiguity). In the case of face tracking, we define $\mathbf{x} = (x, y, s)$, namely the position and size. An observer outputs $p(\mathbf{z}|\mathbf{x})$ for each input candidate $\mathbf{x}$. Since we adopt m observers, it is convenient to define $\mathbf{z} = (z_1, \ldots, z_m)$, and denote the output of the k-th observer as $p(z_k|\mathbf{x})$.

Further, each observer here is represented by its learner L, training sample set S, feature pool F for learning, as well as the time complexities: offline training complexity τ_{off}, online training complexity τ_{on} and testing complexity τ_{test} (defined as the time to calculate $p(z|\mathbf{x})$ for each $\mathbf{x}$). Therefore we formalize the k-th observer as

$$O_k = (L_k, F_k, S_k, \tau_{k,on}, \tau_{k,off}, \tau_{k,test}). \quad (1)$$

Denote the features selected by L_k as $\hat{F}_k$ ($\subset F_k$), and $\hat{S}_k$ as the test samples input to the k-th observer (the output

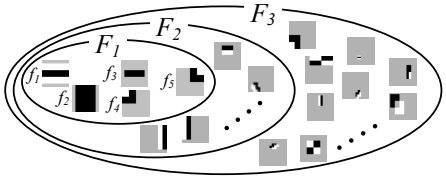


Figure 3. Feature set of each observation model.

k	1	2	3
L_k	A 5-dimension LDA classifier	Discrete AdaBoost on a pool of P LDA classifiers	Real AdaBoost on histogram weak classifiers
F_k	5 pre-selected Haar-like features	50 pre-selected Haar-like features	10,000s of Haar-like features
$\lvert\hat{F}_k\rvert$	5	≤ 50	≈ 500 per view
S_k	Samples of the previous frame	Samples of the previous 5 frames	10,000s of offline samples
$\tau_{k,on}$	$O(\lvert F_1\rvert^2 \lvert S_1\rvert)$	$O(\lvert F_2\rvert^2 \lvert S_2\rvert + \lvert S_2\rvert P^2)$	0
$\tau_{k,off}$	Negligible	Negligible	Several days
$\tau_{k,test}$	$O(\lvert\hat{F}_1\rvert)$	$O(\lvert\hat{F}_2\rvert)$	$O(\lvert\hat{F}_3\rvert)$

Table 1. Our setting of observation models.

should be $p(z_k|\mathbf{x})$ for each $\mathbf{x} \in \hat{S}_k$). The training time $(\tau_{k,off} + \tau_{k,on})$ usually increases with $|F_k|$ and $|S_k|$, and the testing time $\tau_{k,test}$ increases with $|\hat{F}_k|$. And the total offline and online time complexities are

$$\tau_{offline} = \sum_{k=1}^{m} \tau_{k,off}, \tag{2}$$

$$\tau_{online} = \sum_{k=1}^{m} (\tau_{k,on} + |\hat{S}_k| \cdot \tau_{k,test}). \tag{3}$$

For tracking, τ_{online} is of first concern. Notice that both training time $(\tau_{k,on} + \tau_{k,off})$ and testing time $\tau_{k,test}$ increase with an observer's lifespan, since training sample number $|S_k|$ accumulates and a larger F_k (also a more sophisticated L_k) is required to learn the knowledge provided by S_k, and the resulting $\hat{F}_k$ is also larger.

Therefore, to minimize the testing part in (3) it is desirable to arrange the observers in a lifespan-ascending order so that $|\hat{S}_1| < |\hat{S}_2| < \cdots < |\hat{S}_m|$ (Section 3 is devoted to how to achieve this). Meanwhile, to minimize the online training part in (3), F_k must be carefully selected and as much of training as possible should be done offline.

3.2. Configuration and learning

Based on the above analysis, different settings for each observation model is carefully chosen to balance effectiveness and efficiency – in other words, each learner L_k and feature pool F_k should be competent yet not "overqualified" for the learning task on sample set S_k. Table 1 gives a summary.

Feature Sharing (Figure 3) is a distinct feature to be introduced before each observer. We use a Haar-like feature set extended from that of [18]. Calculation of such features is extremely efficient, on the premise that a pyramid of first- and second-order integral images has been built, which is relatively time-consuming in a real-time system. This makes feature sharing a reasonable choice. Feature pool for each observer is selected by offline learning. All observers work on grayscale data.

Observer 1 is an LDA classifier learned on all samples from the previous frame, using only 5 Haar-like features for fast elimination of non-target. Denote the LDA projection vector as $\mathbf{w}$, the 5d feature vector as $\mathbf{f}(\mathbf{x})$ and the classification threshold as η, the observation likelihood is modeled as a sigmoid function based on the classifier's output:

$$p(z_1|\mathbf{x}) \propto \frac{1}{1 + \exp\left(-(\mathbf{w}^T\mathbf{f}(\mathbf{x}) - \eta)\right)}. \tag{4}$$

Observer 2 is a strong classifier boosted from a pool of LDA classifiers. At each frame t, with the samples $S_2 = (S_{2,pos}, S_{2,neg})$ gathered from the past 5 frames, the learning process is

1. Select samples for adding new weak classifiers: all positive samples $S_{2,pos}$ and only the negative samples $S'_{2,neg} = \{\mathbf{x}|\mathbf{x} \in S_{2,neg} \wedge p(z_{t-1,2}|\mathbf{x}) > \xi\}$, where $p(z_{t-1,2}|\mathbf{x})$ is the old observation likelihood and ξ is a threshold. The reason is that new weak classifiers should focus on samples which are not well handled by the old model.
2. Add new weak classifiers by bootstrap. Each is learned by LDA with 10 features selected from F_2.
3. Weight weak classifiers by Discrete AdaBoost [6].
4. Discard weak classifiers which are not selected for a certain number of frames.

Denote the p-th weak classifier as $(\alpha_p, \mathbf{w}_p, \mathbf{f}_p, \eta_p)$, where α_p is the boosted weight, $\mathbf{f}_p$ and $\mathbf{w}_p$ are the features and corresponding projection vector learned by LDA, and η_p is the threshold. The final observation likelihood is modeled by a Sigmoid function of the boosted output:

$$p(z_2|\mathbf{x}) \propto \frac{1}{1 + \exp\left(-\frac{\sum_p \alpha_p \mathbf{sign}\left(\mathbf{w}_p^T\mathbf{f}_p(\mathbf{x}) - \eta_p\right)}{\sum_p \alpha_p}\right)}. \tag{5}$$

Observer 3 is a tree-structured detector similar to [9]. Each tree node is a strong classifier boosted from histogram weak classifiers. Since it consists of multiple layers, for an input $\mathbf{x}$, the output is the number of layers h that $\mathbf{x}$ passes and the confidence c given by the last strong classifier it passed. The observation likelihood is defined as

$$p(z_3|\mathbf{x}) \propto 1/\left(1 + \phi_h \exp(-c)\right), \tag{6}$$

where ϕ_h is the a priori ratio of negative samples to positive samples for the h-th layer (obtained during training process). It decreases as h increases to reflect the fact that the more layers $\mathbf{x}$ passes, the more likely it is the true target.

The elements of learning techniques chosen here are well-established and we would not elaborate on them. However, why they are chosen is paid much consideration and has withstood extensive experiments. *E.g.*, while both Observer 2 and 3 adopt boosting algorithms, the differences between them is meaningful: for Observer 2 we use a much smaller weak classifier pool and Discrete AdaBoost instead of Real AdaBoost, so that both the training time and the risk of over-fitting on small online training set are reduced; on the other hand, since the number of weak classifiers is limited, 10d LDA is adopted to compound simple Haar-like features into more powerful descriptors to guarantee quick training convergence.

4. Cascade particle filter

In this section we first reveal the deficiency of the common approach of particle filter in case of LFR, then describe how the above observers can be coupled tightly together in cascade particle filter to tackle our problem.

4.1. Particle filter under unpredictable motion

By our notation for target state and observation, the aim of a tracking system is to estimate $p(\mathbf{x}_t|\mathbf{Z}_t)$, which stands for the distribution of target state given all observations $\mathbf{Z}_t = (\mathbf{z}_1, \ldots, \mathbf{z}_t)$ up to time t. Particle filter (PF, or CONDENSATION [10]) simulates this distribution by the well-known two-step recursion:

$$\begin{aligned} &\text{Prediction}: p(\mathbf{x}_t|\mathbf{Z}_{t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{Z}_{t-1})d\mathbf{x}_{t-1}, \\ &\text{Update}: p(\mathbf{x}_t|\mathbf{Z}_t) \propto p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{Z}_{t-1}). \end{aligned} \tag{7}$$

Calculation of the integral is carried out by importance sampling, which means samples are generated from a trial distribution. A common practice is to use a presumed $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ as the trail.

But when target motion becomes drastic and unpredictable (*e.g.*, under LFR conditions), such trial distribution will result in gradual departure of the sample set from the true target state which eventually leads to tracking loss (Figure 1(a)). It may be remedied by increasing samples – decreasing efficiency at the mean time. Another choice is to introduce $p(\mathbf{z}_t|\mathbf{x}_t)$ into the trail [14][12], which requires calculation of $p(\mathbf{z}_t|\mathbf{x}_t)$ over large state space. Therefore, under both choices, the system's efficiency degrades towards exhaustive calculation of $p(\mathbf{z}_t|\mathbf{x}_t)$ under unpredictable motion. This problem looms as long as only one computation-intensive $p(\mathbf{z}_t|\mathbf{x}_t)$ is available. So how can we overcome this when multiple observers are available?

4.2. Cascade particle filter

Denote the observation vector as $\mathbf{z} = (z_1, \ldots, z_m)$, assuming independency among the observers will grant us:

$$p(\mathbf{z}|\mathbf{x}) = p(z_1, \ldots, z_m|\mathbf{x}) = \prod_{k=1}^{m} p(z_k|\mathbf{x}). \tag{8}$$

A standard PF can be directly adopted by updating particle weight by $\prod_{k=1}^{m} p(z_k|\mathbf{x})$. But besides what we have analyzed in the above subsection, it is also particularly inefficient here because every particle must be evaluated by every classifier while most of them end up with a weight close to zero (a comparison based on *Effective Sample Size* [13] will illustrate this point in the next section). In the detection problem, the cascade detector has been invented for this very reason. However, the conventional cascade detector can only be viewed as an extreme case where each $p(z_i|\mathbf{x})$ either takes the value 0 or 1, and those $\mathbf{x}$ with $p(\mathbf{z}|\mathbf{x}) > 0$ are classified as positive. And detection using such classifier structure by exhaustive search is like uniformly scattering particles over the whole state space of $\mathbf{x}$.

To overcome the deficiencies of standard PF and cascade detector while combining their merits, our method melts a series of observers into multiple stages of importance sampling (IS). Define

$$\pi_0(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{Z}_{t-1}), \tag{9}$$

$$\pi_k(\mathbf{x}_t) = p(z_{k,t}|\mathbf{x}_t)\pi_{k-1}(\mathbf{x}_t), k = 1..m. \tag{10}$$

By this definition we also have

$$\pi_m(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{Z}_{t-1}) \prod_{k=1}^{m} p(z_k|\mathbf{x}_t) \tag{11}$$

$$= p(\mathbf{x}_t|\mathbf{Z}_{t-1})p(\mathbf{z}_t|\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{Z}_t), \tag{12}$$

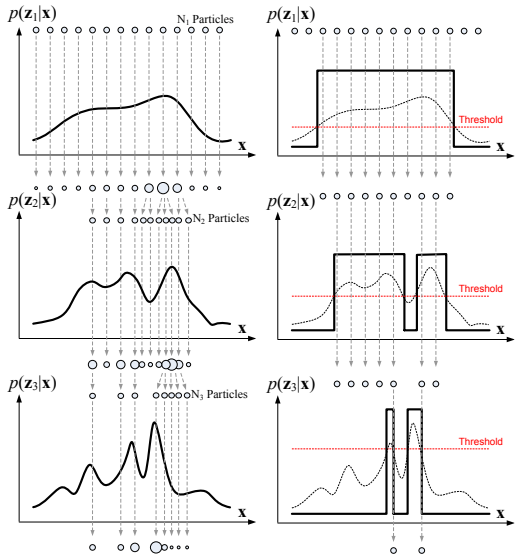
which is the target distribution of our interest.

The algorithm proceeds as follows. At the k-th stage, IS is done to simulate $\pi_k(\mathbf{x}_t)$ with weighted particles. $\pi_{k-1}(\mathbf{x}_t)$ is used as the trial distribution of IS, for which we have already obtained a weighted particle set $P_{k-1,t} = \{\mathbf{x}_{k-1,t}^{(i)}, w_{k-1,t}^{(i)}\}_{i=1}^{N_{k-1}} \sim \pi_{k-1}(\mathbf{x}_t)$ from previous stages. Therefore sampling from this trial distribution can be carried out by re-sampling of $P_{k-1,t}$ to obtain $\{\mathbf{x}_{k,t}^{(i)}, 1/N_k\}_{i=1}^{N_k}$. The weight of $\mathbf{x}_{k,t}^{(i)}$ should be updated according to IS as

$$w_{k,t}^{(i)} = \frac{\pi_k(\mathbf{x}_{k,t}^{(i)})}{\pi_{k-1}(\mathbf{x}_{k,t}^{(i)})} = p(z_{k,t}|\mathbf{x}_{k,t}^{(i)}). \tag{13}$$

The particle set $P_{k,t} = \{\mathbf{x}_{k,t}^{(i)}, w_{k,t}^{(i)}\}$ can be used to represent $\pi_k(x_t)$. IS is repeated for m stages to obtain $P_{m,t} = \{\mathbf{x}_{m,t}^{(i)}, w_{m,t}^{(i)}\} \sim \pi_m(x_t) = p(\mathbf{x}_t|\mathbf{Z}_t)$.

Figure 4 gives an illustration of a cascade PF and a conventional cascade detector both with 3 classifiers. And a



(a) Cascade Particle Filter (b) Cascade Detector

Figure 4. **Illustration of our method and the cascade detector**, both with 3 classifiers. Here we assume the initial particles for the cascade particle filter is uniformly distributed, which might not be the exact case in practice due to temporal propagation.

Method	Time Complexity	Comments
Standard Particle Filter	$N\sum_{k=1}^{m}\tau_{k,test}$	When N is sufficiently large to ensure an accurate result, the time complexity also becomes much larger than the other two methods.
Cascade Detector	$\sum_{k=1}^{m}N'_k\tau_{k,test}$	$N'_1 > N'_2 > \cdots > N'_m$. N'_k depends on the fixed threshold of each classifier. Candidates are selected uniformly over the state space.
Our method	$\sum_{k=1}^{m}N_k\tau_{k,test}$	$N_1 > N_2 > \cdots > N_m$. N_k acts as a dynamic "threshold" in the sense of controlling how many particles should be kept in each stage. Particles are re-distributed and re-weighted in each stage to adapt to the target distribution.

Table 2. A comparison among standard particle filter, conventional cascade detector, and our method. $\tau_{k,test}$ is the cost of calculating $p(z_k|\mathbf{x})$; N_k(or N'_k) is the number of particles (or passed candidates) in the k-th stage.

further comparison of standard PF, cascade detector and our method is shown in Table 2.

In implementation, we have observed $p(z_k|\mathbf{x})$ noisy and with many peaks (see Figure 5(a) for an idea), which is typical for discriminative models. And the peaks of successive observers do not necessarily overlap. This would cause problems in approaches using observation model in a cascade detector manner without re-distributing and re-weighting particles (such as that in [20] which simply discards low-weight particles and keeps high-weight ones). But in our framework it is handled conveniently by adding

With $\{\mathbf{x}^{(i)}_{m,t-1}, w^{(i)}_{m,t-1}\}_{i=1}^{N_m}$ the particle set at the previous time step, proceed as follows at time t:

- **Resample** (also to enlarge the particle number from N_m to N_1): simulate $\alpha_j \sim \{w^{(i)}_{m,t-1}\}_{i=1}^{N_m}$, and replace $\{\mathbf{x}^{(i)}_{m,t-1}, w^{(i)}_{m,t-1}\}_{i=1}^{N_m}$ with $\{\mathbf{x}^{(\alpha_j)}_{m,t-1}, 1/N_1\}_{j=1}^{N_1}$.
- **Prediction**: For $i = 1..N_1$, simulate $\mathbf{x}^{(i)}_{1,t} \sim p(\mathbf{x}_t|\mathbf{x}^{(i)}_{t-1})$, note that $\mathbf{x}^{(i)}_{1,t}$ should be diffused enough to cover the expected range of possible state transition.
- **At stage** 1 **do**: For $i = 1..N_1$ let $w^{(i)}_{1,t} = p(z_{1,t}|\mathbf{x}^{(i)}_{1,t})$.
- **For stage** $k = 2..m$ **do**:
  - Resample (also to reduce the particle number from N_{k-1} to N_k): simulate $\alpha_j \sim \{w^{(i)}_{k-1,t}\}_{i=1}^{N_{k-1}}$ and replace $\{\mathbf{x}^{(i)}_{k-1,t}, w^{(i)}_{k-1,t}\}_{i=1}^{N_{k-1}}$ with $\{\mathbf{x}^{(\alpha_j)}_{k-1,t}, 1/N_k\}_{j=1}^{N_k}$.
  - Add small diffusion: For $i = 1..N_k$, simulate $\mathbf{x}^{(i)}_{k,t} \sim g(\mathbf{x}_{k,t}|\mathbf{x}^{(i)}_{k-1,t})$, and let $\lambda^{(i)} = g(\mathbf{x}^{(i)}_{k,t}|\mathbf{x}^{(i)}_{k-1,t})$, where g is a 0-mean gaussian.
  - For $i = 1..N_k$, let $w^{(i)}_{k,t} = p(z_{k,t}|\mathbf{x}^{(i)}_{k,t})/\lambda^{(i)}$.
  - Normalize weight so that $\sum_{i=1}^{N_k} w^{(i)}_{k,t} = 1$.
- **Estimate** $\mathbf{x}_t$: Cluster the particles $\{\mathbf{x}^{(i)}_{m,t}, w^{(i)}_{m,t}\}_{i=1}^{N_m}$. Select the cluster C with maximum weight. Output $\hat{\mathbf{x}}_t = \frac{\sum_{i\in C} w^{(i)}_{m,t}\cdot\mathbf{x}^{(i)}_{m,t}}{\sum_{i\in C} w^{(i)}_{m,t}}$.

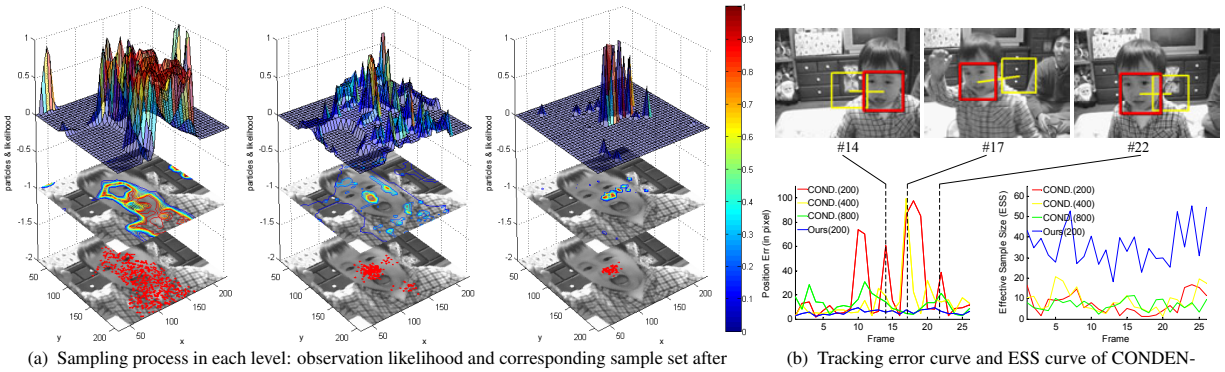
Table 3. Algorithm of the cascade particle filter.

a small gaussian diffusion into the trial distributions in each IS stage. The final algorithm is show in Table 3.

Also, we would like to mention the annealed particle filter [5], which is quite similar to our approach at the first glance. Both of them seek for the global maximum of multi-modal likelihood functions in a particle-based stochastic manner. However, they are different in motive and measure: 1) annealed PF is originally designed for searching in a high dimensional state space; cascade PF is for searching in a large range of state space; 2) the number of particles in each stage of cascade PF is decreased instead of being constant as in annealed PF; 3) different observers are adopted in each stage of cascade PF, while annealed PF uses the same likelihood function only with a increasing power $(w(\mathbf{z},\mathbf{x})^{\beta_m}$ for the m-th stage).

5. Experiment and discussion

The algorithm is implemented in C++ and runs at about 30fps for 320x240 video (single target) on a PC with Pentium 2.8GHz CPU. The numbers of particles adopted in each level are 3000, 600, and 200 respectively.



(a) Sampling process in each level: observation likelihood and corresponding sample set after updating weight.

(b) Tracking error curve and ESS curve of CONDENSATION and proposed approach. Results of proposed approach are shown in frames where CONDENSATION have large error (yellow rectangle denotes target in the previous frame).

Figure 5. Illustration of sampling process, comparison of tracking accuracy and sampling efficiency between CONDENSATION and proposed approach.

5.1. Analysis of the sampling process

To give a vivid view of how our algorithm works, Figure 5(a) shows the sampling steps in one frame of a video. As is expected, likelihood output by the higher-level observer is more peaked, and as a result the particles are more focused around the true target state after each level's re-weighting and re-sampling. Also notice that the likelihood output of the later two observation models are not smooth even near the true target (which is typical of a discriminative model), making the re-sampling with diffusion necessary.

With comparison to CONDENSATION with different number of particles, Figure 5(b) includes a quantitative analysis of sampling effectiveness and efficiency, by the curve of tracking error on the left and the curve of *Effective Sample Size* (ESS) on the right. It is obvious that enlarging particle set of CONDENSATION compensates its poor accuracy under drastic motion to some extent (compare the green curve with 800 particles and the red one with 200 particles), but the error of our method is still lower (the blue curve). Further, a much higher sampling efficiency of our method is shown by the "rule of thumb" of importance sampling [13] – the ESS, calculated by $ESS(N) = N/(1+cv^2(\omega))$, where N is particle number and $cv^2(\omega)$ is the coefficient of variation of the unnormalized weight. ESS can be interpreted as that N weighted samples are worth of $ESS(N)$ samples drawn from the target distribution.

5.2. Quantitative comparison

A comparison is done among CONDENSATION, mean shift using color histogram (implemented by [1]), online tracker of [19] (by online selecting Haar-like features) and our method on videos with manually labeled ground truth for comparison. All videos are taken by hand-held camera and down-sampled to 5pfs. `baseball.mpg` (Figure 7) and `hopping.mpg` record single person engaged in sports, `excursion{1|2}.mpg` (Figure 1 and 6) record several people walking along a passage, and `boy{1|2}.mpg` (Figure 8) record children playing (2676 frames in all). By using these videos we aim to test our algorithm on concurrence of abrupt motion, low frame rate and a moving camera. Tracking accuracy is shown in Figure 9 and Table 4. Our method shows much higher accuracy than other methods under LFR condition. For more tracking results please refer to the supplementary video.

5.3. Discussion on scenarios

Online and offline fusion. In Figure 7 we selected two challenging cases. The first is a fast camera motion which caused both drastic target motion and appearance blur. Applying the offline trained face detector, we observe missing detection. Moreover, an offline trained model cannot identify targets by online context. On the other hand, our method successfully grabs the target appearance and adapts itself to the variation (blur). The second case is a quick pose change (more than 120° within 5 frames). While our tracker accurately located the target, a tracker which solely depends on online knowledge exhibits a "drift" which eventually leads to failure. These phenomena are frequently observed in our experiments, which illustrate the importance of complementing online and offline models with each other.

Multi-target tracking. We extend the proposed method to multi-target tracking in a simple way: after each target is tracked individually, several particle clusters are obtained for each target. A data-association stage is then added, by greedily associate one target with the highest-weighted cluster, and then eliminate clusters of un-determined targets that overlaps with determined ones. During the experiment, we observe that the online observation models not only develop discriminative power against background clutter, but

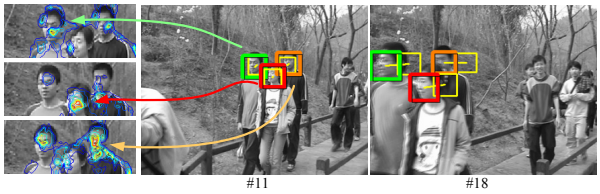


Figure 6. Multi-target tracking and likelihood output of online observers for different targets, yellow rectangle denotes target in the previous frame (`excursion2.mpg`).

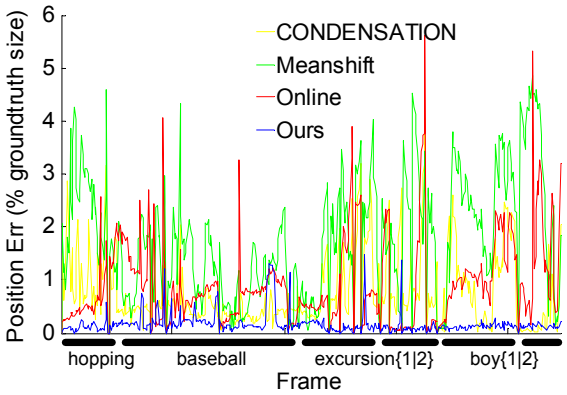


Figure 9. Position error curves of different methods.

Algorithm	Mean Pos Err	Mean Scale Err	Success Rate
CONDENSATION	0.7592	0.2654	48.65%
Mean shift (color)	1.7808	0.3802	8.74%
Online tracker in [19]	0.9167	0.2869	32.06%
Ours	**0.1641**	0.2079	**94.39%**

Table 4. Comparison with other methods. Error is normalized by ground truth target size. And the criterion of successful tracking is that both position and scale errors are smaller than 0.5.

also against ambiguity with other targets (Figure 6).

6. Conclusion and future work

There are always two essential parts in a tracking system: the observation model, and the temporal inference process based on it. To deal with LFR conditions, we have pushed both of them towards integration with their counterparts in a detection problem, to enhance discriminative power and efficiency. A temporal probabilistic combination of observers is presented, in which each observer is statistically learned from online or offline samples, to grasp target appearance of longer or shorter temporal vicinity. The cascade particle filter serves for efficient fusion, and combines the advantages of a cascade detector and the nonparametric temporal propagation framework of particle filter.

Using multiple observers has enabled efficient cascade structure and avoided the difficulty in incrementally updating one offline learned model, which could be time-consuming when the model complexity is high (especially for discriminative models); also, setting the updating ratio can be very subtle because over-update may even ruin the original model. Nevertheless, we regard sustainable incremental learning as a promising direction for further improving our observers. Also, a detailed performance analysis by modeling and controlling training and testing errors of online learned observers will be necessary to make the system more robust and stable.

7. Acknowledgement

This work is supported in part by National Science Foundation of China under grant No.60332010, No.60673107, National Basic Research Program of China under Grant No.2006CB303100, and it is also supported by a grant from Omron Corporation.

References

[1] Intel opencv library. http://www.sourceforge.net/projects/opencvlibrary.

[2] S. Avidan. Ensamble tracking. *PAMI*, 29(2):261–271, 2007.

[3] S. Birchfield. Source code of the klt feature tracker. http://www.ces.clemson.edu/~stb/klt/, 2006.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, 2000.

[5] J. Deutscher, A. Blake, and I. Reid. Experiments with a new boosting algorithm. In *CVPR*, 2000.

[6] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, 1996.

[7] M. Han, A. Sethi, W. Hua, and Y. Gong. A detection-based multiple object tracking method. In *ICIP*, 2004.

[8] G. Hua and Y. Wu. Multi-scale visual tracking by sequential belief propagation. In *CVPR*, 2004.

[9] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *ICCV*, 2005.

[10] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *IJCV*, 28(1):5–28, 1998.

[11] R. Kaucic, A. G. A. Perera, G. Brooksby, J. Kaufhold, and A. Hoogs. A unified framework for tracking through occlusions and across sensor gaps. In *CVPR*, 2005.

[12] C. Liu, H.-Y. Shum, and C. Zhang. Hierarchical shape modeling for automatic face localization. In *ECCV*, 2002.

[13] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, New York, 1994 (ISBN: 0-387-95230-6).

[14] K. Okuma, A. Taleghani, D. Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, 2004.

[15] F. Porikli and O. Tuzel. Object tracking in low-frame-rate video. *SPIE Image and Video Communications and Processing*, 5685:72–79, 2005.

[16] J. Sullivan, A. Blake, M. Isard, and J. MacCormick. Object localization by bayesian correlation. In *ICCV*, 1999.

[17] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991.

[18] P. Viola and M. Jones. Robust real-time object detection. In *IEEE Workshop on Statistical and Theories of Computer Vision*, 2001.

[19] J. Wang, X. Chen, and W. Gao. Online selecting discriminative tracking features using particle filter. In *CVPR*, 2005.

[20] C. Yang, R. Duraiswami, and L. Davis. Fast multiple object tracking via a hierarchical particle filter. In *ICCV*, 2005.

(a) Offline face detection. It fails due to motion blur, and is unable to identify different persons.

#56 #57 #58 #59 #60 Target Trajectory

(b) Proposed approach under drastic camera motion and considerable blur (yellow rectangle denotes target in the previous frame).

(c) Online tracking by method in [19], drift can be observed.

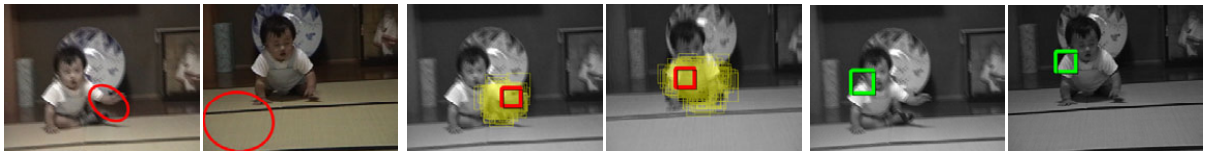
#24 #25 #26 #27 #28 Target Trajectory

(d) Proposed approach under fast view change (right full profile to left profile in 5 frames).

Figure 7. Comparison with offline approach and online approach in challenging cases (`baseball.mpg`).

#59 #64 #86 #99 #101 #105

(a) Tracking by proposed approach under poor illumination, significant camera motion and fast zooming (yellow rectangle denotes target in the previous frame).



#59 #64

(b) Mean Shift by color histogram.

#59 #62

(c) CONDENSATION.

#59 #64

(d) Online tracking by method in [19].

#18 #26 #27 #31 #33 #38

(e) Tracking a jumping and running kid in a cluttered scene by proposed approach (yellow rectangle denotes target in the previous frame).

#31 #33

(f) Mean Shift by color histogram.

#27 #28

(g) CONDENSATION.

#28 #31

(h) Online tracking by method in [19].

Figure 8. Comparison with other methods on 5fps videos under various conditions (`boy1.mpg, boy2.mpg`).