CamShift Guided Particle Filter for Visual Tracking

Zhaowen Wang, Xiaokang Yang, Yi Xu and Songyu Yu
Institute of Image Communication and Information Processing
Shanghai Jiao Tong University, Shanghai, PRC 200240
E-mail: {whereaswill,xkyang, xuyi, syyu}@sjtu.edu.cn

***Abstract*— Particle filter and mean shift are two important methods for tracking object in video sequence, and they are extensively studied by researchers. As their strength complements each other, some effort has been initiated in [1] to combine these two algorithms, on which the advantage of computational efficiency is focused. In this paper, we extend this idea by exploring even more intrinsic relationship between mean shift and particle filter, and propose a new algorithm, CamShift guided particle filter (CAMSGPF). In CAMSGPF, two basic algorithms - CamShift and particle filter - can work cooperatively and benefit from each other, so that the overall performance is improved and some redundancy in algorithms can be removed. Experimental results show that the proposed method can track objects robustly in complex environment, and is much faster than the existing methods.**

I. Introduction

Tracking visual objects through image frames has been a fundamental topic in computer vision field and is widely applied to surveillance, robotics, human machine interface, object based video coding, etc. However, the task of robust tracking is challenging regarding fast motion, occlusion, structural deformation, illumination variation, background clutters, real-time restriction, etc.

To handle these problems, many efforts have been paid to devise good tracking algorithms. One promising category is sequential Monte Carlo methods, also known as particle filers, which estimate the most likely posterior with discrete sample-weight pairs in a Bayesian Network frame. The basic idea is introduced by Hammersley et al in [2], and is implemented into various improved versions over the last decade. Due to particle filters' non-Gaussian non-linear assumption and multiple hypothesis property, they are successfully applied to visual tracking, e.g. in [3], [4], and show extra merits in cluttered environment. However, the inefficiency in sampling and the huge computational complexity limit the usefulness of particle filter in on-line tracking.

Another popular tracking method is mean shift procedure, which finds the local maximum of probability distribution in the direction of gradient. Comaniciu et al. propose a mean shift based tracking method in [5]. Bradski [6] extends it to CamShift by adaptively changing the scale of search window. As a deterministic method, mean shift keeps single hypothesis and thus is computationally efficient. But it may run into trouble when similar objects are presented in background or when occlusion happens.

Based on the pros and cons of particle filter and mean shift, Shan et al. [1] proposed a new algorithm, the Mean Shift Embedded Particle Filter (MSEPF), to integrate the advantages of the two methods for object tracking. In MSEPF, mean shift is performed on each of the particles after they are propagated, so that the particles are "herded" to nearby local modes with large weights. In this way, the posterior can be better estimated even with a smaller sample set, and the computation complexity of particle filter is reduced proportionally.

In MSEPF, mean shift is just used as a subsidiary tool to draw better samples for particle filter. In fact, we find particle filter can also play a role in assisting the mean shift procedure. Therefore, in this paper, we extend the idea of MSEPF and propose a novel algorithm, CamShift Guided Particle Filter (CAMSGPF). In CAMSGPF, CamShift facilitates particle filter in drawing a good sample set as it is in MSEPF; and in return, particle filter helps to improve the scale estimation and simplify the complexity of CamShift. In this way, CamShift and particle filter can work together coherently. The proposed CAMSGPF algorithm is efficient and robust, and surpasses other particle filters and mean shift based trackers in general.

The remainder of the paper is organized as follows. We present in detail how CAMSGPF integrates CamShift into particle filter in Section II, and then show experimental results in Section III, and finally conclude the paper in Section IV.

II. CamShift Shift Guided Particle Filter

The main frame of CAMSGPF proposed in this paper is based on the well-known Sequential Importance Resampling (SIR) [7] particle filter, with CamShift integrated. In the algorithm, the state probability distribution of the target is estimated via a finite set of N samples (particles) with state $\left\{x^i\right\}_{i=1\ldots N}$. Given the sample set at the previous time, $\left\{x_{t-1}^i\right\}_{i=1\ldots N}$, the CAMSGPF starts by propagating each sample with a stochastic displacement according to dynamic model. The resulting samples $\left\{\tilde{x}_t^i\right\}_{i=1\ldots N}$ are further shifted by a modified version of CamShift, so that the new sample set, $\left\{\bar{x}_t^i\right\}_{i=1\ldots N}$, will be more close to distribution modes. Then we assign weight $\left\{w_t^i\right\}_{i=1\ldots N}$ to each sample by evaluating their likelihood against observation z_t at the new state and taking into account the biasing effect of CamShift previously applied. Finally, a resample step multiplies samples with large weight and generates an unweighted sample set $\left\{x_t^i\right\}_{i=1\ldots N}$, which corresponds to the target distribution at current time. In this way, CAMSGPF can recursively find the latest target state. The flowchart of CAMSGPF is illustrated in Fig. 1.

In CAMSGPF, CamShift is adopted for active scale adaptation. And more importantly, the cooperative relationship

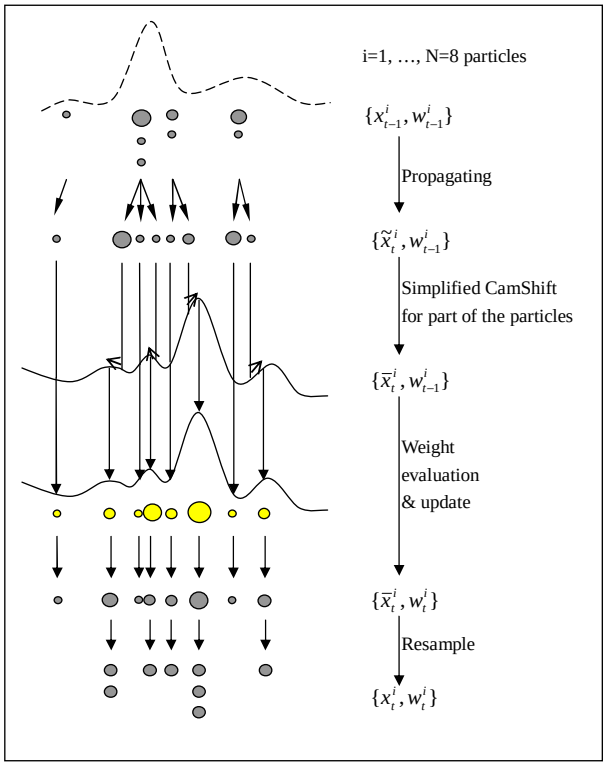

Fig. 1. The framework of CAMSGPF. Each blob's position and area represent the state and weight of a sample. The highlighted color signifies weights evaluated directly from observation model.

between particle filter and CamShift is fully exploited. On one hand, the particles are concentrated by CamShift to higher probability nearby modes, in terms of both the position and scale of the target. On the other hand, CamShift can achieve a *better scale adaptation* than being used alone with particle filter's multi-hypothesis nature; and CamShift can be applied in a much simplified way to further *boost the algorithm efficiency* due to the 2nd AR dynamic model prediction. In the following paragraphs, we shall explain in detail how these points are attained in CAMSGPF.

A. System Models

In our system, the target to be tracked in video sequence is modeled by a state vector:

$$x = (x_c, y_c, w, h)^T \tag{1}$$

so that a rectangle box centered at coordinate (x_c, y_c) with width w and height h just covers the target area. The dynamic of state transition corresponds to a standard 2nd order autoregressive process:

$$x_t = x_{t-1} + (x_{t-1} - x_{t-2}) + n_t \tag{2}$$

That is, the current state x_t is predicted as the sum of three terms: the previous state at time $t-1$, the displacement of last transition, and a gaussian noise, n_t. This simple model can well simulate most ordinary object motions, and plays a vital role in simplifying the overall algorithm as can be seen in later discussion.

Following [4], we use HSV color histogram to build the observation model. Given the current observation z_t (i.e. the current image frame), the candidate color histogram $q(x_t)$ is calculated on z_t in the region specified by x_t. Then it is compared with the reference color histogram q^* by Bhattacharyya similarity metric $D[,]$, resulting to the likelihood distribution:

$$p(z_t|x_t) \propto e^{-\lambda D^2[q^*, q(x_t)]} \tag{3}$$

where λ is set to 20 for most applications.

B. Efficient Position Shift

In CAMSGPF, the samples $\{\tilde{x}_t^i\}_{i=1\ldots N}$ drawn according to state dynamic Eq. (2) are first shifted in their position subspace by mean shift vector [5] for a few iterations. Let's denote the current position of a sample x as $p \equiv [x_c, y_c]^T$, then its new position after one iteration will be:

$$p' = \frac{\sum_{i=1}^{M} a_i w(a_i) g(\left\| \frac{p-a_i}{h} \right\|^2)}{\sum_{i=1}^{M} w(a_i) g(\left\| \frac{p-a_i}{h} \right\|^2)} \tag{4}$$

where $\{a_i\}_{i=1\ldots M}$ are pixel coordinates within the rectangle area specified by state x, $w(a_i)$ is the weight indicating the ratio of histogram bin values corresponding to a_i in the current and reference color histogram. $g()$ is a kernel profile function, and h is window radius to normalize the coordinate a_i.

In MSEPF, the mean shift iteration in Eq. (4) is applied on every sample in sample set, and goes on until convergence or maximum number of iterations is reached. It is claimed in [1] this will greatly reduce the time consumption on particle filter, because fewer particles are required to accomplish tracking. Unfortunately, this improvement in efficiency will be partly canceled out by the introduction of mean shift, which will take extra time for the whole algorithm. What's worse, a traditional mean shift procedure [5] is more complicated than a generic particle filter recursion cycle, so each particle in MSEPF will spend most of its time in the mean shift step.

To further improve the algorithm efficiency, we must simplify the mean shift procedure in CAMSGPF. This is approached in the following three ways:

1) Mean shift is applied only on some particles randomly selected.
2) Only a small and fixed number of iterations are carried out on each sample.
3) Unlike the original mean shift procedure, which checks the correctness of mean shift vector in each iteration, we omit it at all in CAMSGPF.

Because the total time consumed on mean shift is proportional to the number of particles subjected to it and the number of iterations each individual mean shift undergoes, our simplification will boost the algorithm speed substantially.

Moreover, it can be found that, in the context of particle filtering, the simplifications above will have little influence on the performance of mean shift. The reason lies in the displacement term of Eq. (2). Once a particle is shifted, the

effect of mean shift will be built into the displacement term and carried to the future transitions. Then even if a particle is not shifted at every time step, or is not shifted thoroughly to the maximum nearby mode in a restricted number of iterations, the particle will still move in the direction to the maximum "inertially" as long as the distribution does not change dramatically.

Furthermore, with the resampling step of particle filter, there is no need to worry about the correctness of mean shift in each iteration. In CAMSGPF, badly shifted particles will be removed by resampling, and particles shifted towards local modes will be resampled more than once. In this way, particle filter performs the evaluation on mean shift iterations and further simplifies the mean shift procedure.

Now we can safely apply the simplified mean shift on part of the samples at each time step. To keep every particle being mean-shifted evenly, once for a while, we sort the samples according to their weights and determine whether to apply mean shift or not by their modular-ed index:

$$\begin{cases} MeanShift^*(x^i, I) & \text{if } i\%N_s < n_1 \\ \text{No Mean Shift} & \text{otherwise} \end{cases} \tag{5}$$

where i is the index of the sorted particles, N_s is a fraction of particle number N, and n_1 is an integer usually taken to be half of N_s. We have used $MeanShift^*(,)$ to denote the simplified mean shift, which takes argument x^i as the initial search position, and I as the number of iterations. As particles' weights are changed dynamically, this operation will guarantee all the samples a fair chance to get mean-shifted.

C. Adaptive Scale Adjustment

After the samples in $\{\tilde{x}_t^i\}_{i=1\ldots N}$ are mean-shifted in position subspace, they are further refined in scale subspace using CamShift algorithm in CAMSGPF. As noted in [6], CamShift extends the mean shift algorithm so that the size of the searching window can be adjusted to fit the changing scale of the target. The calculation is based on pixel's likelihood weight $w(a_i)$, which is the byproduct in evaluating mean shift vector according to Eq. (4). Since a denser pixel weight distribution (larger zeroth moment of the likelihood distribution image) implies a larger target size, the window size is estimated empirically in [6] as a function of the zeroth moment:

$$l = k\sqrt{\frac{M_{00}}{256}} \tag{6}$$

where l is the width or height of the window, k is a constant. And the zeroth moment M_{00} of the corresponding rectangle window area of the current state x is calculated by

$$M_{00} = \sum_{i=1}^{M} w(a_i) \tag{7}$$

where $\{a_i\}_{i=1\ldots M}$ and $w(a_i)$ are defined as they are in Eq. (4). The moment should be first normalized by the maximum value of probability distribution (256 in the 8-bit case).

Since the relationship between zeroth moment and target size in Eq. (6) is found empirically, it is practical only in some specific cases. For example, with parameter setting in [6], CamShift can track human face successfully, but may fail to adjust the window size properly in more general applications (Fig. 2).

(a) (b)

Fig. 2. With the moment-size relationship of Eq. (6), CamShift succeeds in tracking a man's face (a); while the scale adaptation fails when tracking a red car in field (b).

In CAMSGPF, we apply CamShift with different moment-size-estimation functions on different particles. Some of these functions tend to scale the particles' size up or down, while others do not change the size at all (in this case, CamShift degrades to mean shift). After doing some manipulations on Eq. (6), we can obtain two scale factors used in our method, s_u for scaling up and s_d for scaling down:

$$s_u = k_1\sqrt{\frac{M_{00}}{256 \times w \times h}} \tag{8a}$$

$$s_d = k_2\sqrt{\frac{M_{00}}{256 \times w \times h}} \tag{8b}$$

where w and h are the width and height component of the current sample state x; k_1 and k_2 are constants satisfying $k_1 > k_2$. The scale subspace of the state x is thus updated to be:

$$w \leftarrow s \times w \tag{9a}$$

$$h \leftarrow s \times h \tag{9b}$$

where s is the scaling factor. It is randomly selected from s_u, s_d and 1 for each particle, so that different scales are tried on different particles. Then after these particles are evaluated using observation model, the better scaled ones can be picked out, while some inappropriately scaled ones will be eliminated by resampling. As w and h can both increase and decrease in the scale subspace, they will eventually settle on the exact size of the target. With this multi-hypothesis and test paradigm, we can adjust the size to the best without knowing the precise moment-size-estimation function.

To ensure that all the particles will have equal opportunity to try on different choices of scales, we employ the technique similar to Eq. (5). The scaling factor s is selected for a sorted particle x^i as:

$$s = \begin{cases} s_u & \text{if } i\%N_s = n_2 \\ s_d & \text{if } i\%N_s = n_3 \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

where parameters n_2 and n_3 can be any integers less than n_1 in Eq. (5), as particles to be scaled must be mean-shifted at

first hand. And in our implementation they are taken as 1 and 2 for simplicity. The physical meaning of N_s is also clear from Eq. (10): it is defined such that each one out of N_s particles will be scaled up and down.

To combine Eq. (5) and Eq. (10) in a more compact form, let $CamShift^*(,,)$ (to be elaborated in Table II later) be the concatenation of the simplified mean shift $MeanShift^*(,)$ and a subsequent CamShift scaling, and the selective mean-shifting and scaling mechanism can be rewritten in the form:

$$\begin{cases} \text{No CamShift} & \text{if } i\%N_s \geq N_s/2 \\ CamShift^*(x^i, I, +1) & \text{if } i\%N_s = 1 \\ CamShift^*(x^i, I, -1) & \text{if } i\%N_s = 2 \\ CamShift^*(x^i, I, 0) & \text{otherwise} \end{cases} \tag{11}$$

where the 3rd argument in $CamShift^*(,,)$ signals the choice of scaling factor.

So far, we can find that in CAMSGPF, the function of CamShift is to give the particles some crude implications on the direction to propagate (for both position and scale), while the judgment and feedback is left to particle filter. It is because of this interactive relationship that we name this algorithm CamShift "guided" particle filter.

D. Weight Evaluation

Having CamShift-ed samples from $\{\tilde{x}_t^i\}_{i=1...N}$ to $\{\bar{x}_t^i\}_{i=1...N}$, we are ready to update their weights $\{w_t^i\}_{i=1...N}$ at the new states. The new weight of each sample is found as follows:

$$w_t^i \propto w_{t-1}^i \frac{p(z_t|\bar{x}_t^i)p(\bar{x}_t^i|x_{0:t-1}^i)}{q(\bar{x}_t^i|x_{0:t-1}^i, z_t)} \tag{12}$$

where $q(\bar{x}_t^i|x_{0:t-1}^i, z_t)$ is the proposal distribution, $p(z_t|\bar{x}_t^i)$ is given by Eq. (3), and prior distribution $p(\bar{x}_t^i|x_{0:t-1}^i)$ can be derived from state dynamic Eq. (2) to be:

$$p(\bar{x}_t^i|x_{0:t-1}^i) = \mathcal{N}(2x_{t-1}^i - x_{t-2}^i, \sigma) \tag{13}$$

where $\mathcal{N}(,)$ denotes Gaussian distribution, σ is the covariance.

The difficulty to evaluate Eq. (12) lies in finding the proposal $q()$. Before CamShift is embedded into particle filter, $q()$ just takes the same form as the prior Eq. (13). However, in CAMSGPF, the effect of CamShift should also be taken into account; otherwise, the posterior estimated by $\{\bar{x}_t^i, w_t^i\}_{i=1...N}$ would be biased. We can view CamShift as a subsequent sample drawing step conditioned on the samples $\{\tilde{x}_t^i\}_{i=1...N}$, which is drawn beforehand according to the prior distribution. Then the new proposal distribution becomes:

$$q(\bar{x}_t^i|x_{0:t-1}^i, z_t) = \int \bar{p}(\bar{x}_t^i|\tilde{x}_t^i, z_t)p(\tilde{x}_t^i|x_{0:t-1}^i)d\tilde{x}_t^i \tag{14}$$

where $\bar{p}(\bar{x}_t^i|\tilde{x}_t^i, z_t)$ is the probability that state $\tilde{x}_t^i$ will be CamShift-ed to $\bar{x}_t^i$ given the observation z_t. As CamShift is deterministic, the value of $\bar{p}()$ is either 1 or 0, and can be determined explicitly by checking whether the result of CamShift-ing $\tilde{x}_t^i$ by Eq. (11) will collide with $\bar{x}_t^i$. However, doing CamShift on all possible $\tilde{x}_t^i$ is time-exhausting, in practice we use a Gaussian model similar to [8] to approximate $\bar{p}()$:

$$\bar{p}(\bar{x}_t^i|\tilde{x}_t^i, z_t) \approx \mathcal{N}(\tilde{x}_t^i, \Sigma) \tag{15}$$

where Σ is a diagonal covariance matrix. As CamShift will not shift a sample too far away from its initial state, the approximation in Eq. (15) can counterbalance CamShift's biasing on posterior to some extent.

E. Summary of CAMSGPF

For clarity, we briefly encapsulate the overall CAMSGPF algorithms in Table I, which mathematically describes the flowchart in Fig. 1.

TABLE I
ALGORITHM OF CAMSHIFT GUIDED PARTICLE FILTER

$\{x_t^i, w_t^i\}_{i=1...N} = CAMSGPF(\{x_{t-1}^i, w_{t-1}^i\}_{i=1...N})$

- for $(i = 1 : N)$
  - Propagate particle x_{t-1}^i by Eq. (2) to get $\tilde{x}_t^i$
  - Selectively CamShift $\tilde{x}_t^i$ to $\bar{x}_t^i$ according to Eq. (11)
  - Evaluate the observation likelihood $p(z_t|\bar{x}_t^i)$ by Eq. (3)
  - Update weight w_t^i by Eq. (12)
- endfor
- Sort $\{\bar{x}_t^i, w_t^i\}_{i=1...N}$ according to weight w_t^i
- Resample $\{\bar{x}_t^i, w_t^i\}_{i=1...N}$, producing un-weighted sample set $\{x_t^i, 1/N\}_{i=1...N}$

And the modified $CamShift^*(,,)$ algorithm in Eq. (11) is shown in Table II.

TABLE II
ALGORITHM OF THE MODIFIED CAMSHIFT

$\bar{x} = CamShift^*(\tilde{x}_0, I, \epsilon)$

- Set the current search window to the rectangle represented by $\tilde{x}_{j-1}$
- Evaluate all the pixel weight $w(a_i)$ inside the window
- Shift the position component $(x_c, y_c)^T$ of $\tilde{x}_{j-1}$ by Eq. (4) :
- if $(\epsilon = +1)$ Find scaling factor s by Eq. (8a)
  elseif $(\epsilon = -1)$ Find scaling factor s by Eq. (8b)
  else set $s = 1$
- Update the scale component w and h of $\tilde{x}_{j-1}$ using Eq. (9a), (9b)
- if $(j = I)$ stop, set $\bar{x} = \tilde{x}_j$
  else j++, and return to the first step

III. EXPERIMENTAL RESULTS

In this section, the performance of CAMSGPF is compared with other trackers in a number of aspects. In the experiments, the parameters of CAMSGPF are set as follows: $k_1 = 1.2$; $k_2 = 1$; $N_s = 10$; $I = 2$. All the tests are carried out on 320×240-pixel sequences with a Pentium IV 2.8G PC.

We compare the general tracking ability of different algorithms by examining the minimal number of particles required by each of them to barely achieve successful tracking. From the experiments on "hockey" sequence (shown in Fig. 3, tracking results by CAMSGPF only), we observe that this number is 35 at least for generic particle filter, while 10 for both of MSEPF and CAMSGPF. It is clear that mean shift or

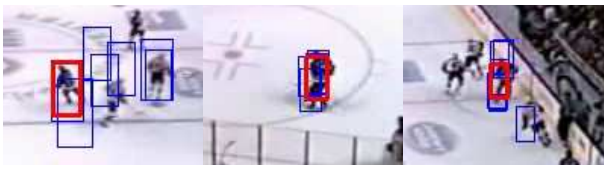

Fig. 3. "hockey" sequence successfully tracked by CAMSGPF with 10 particles (frame 56, 211, 429)

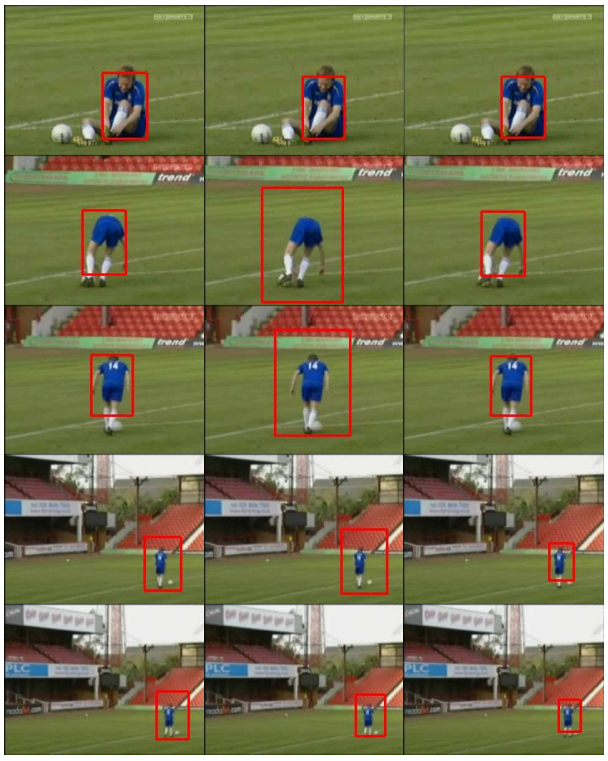

Fig. 4. "soccer" sequence tracked by MSEPF, CamShift and CAMSGPF (in the columns from left to right). (frame 114, 123, 176, 193)

CamShift can help the particle filter a lot, reducing the number of particles by 71%.

The ability of scale adaptation is tested in the "soccer" sequence, as shown in Fig. 4. The tracking result of CAMSGPF is displayed in the third column, in comparison with those tracked by MSEPF and CamShift (with moment-size-estimation functions tuned to this sequence), in the first and second column. Our method turns out to adapt to scale change of the target best, since the boundary box best matches the target size.

To validate the significance of applying the simplified CamShift on part of the particles, we compare the time consumption of CAMSGPF with MSEPF on several sequences. Both of the algorithms can track the targets correctly most of the time, as Fig. 5(d) demonstrates. The average time consumed per frame by the two is plotted in Fig. 5 (a)-(c) correspondingly, with regard to different number of particles used. For all tests, the time consumption grows linearly with the number of particles, and CAMSGPF is consistently faster

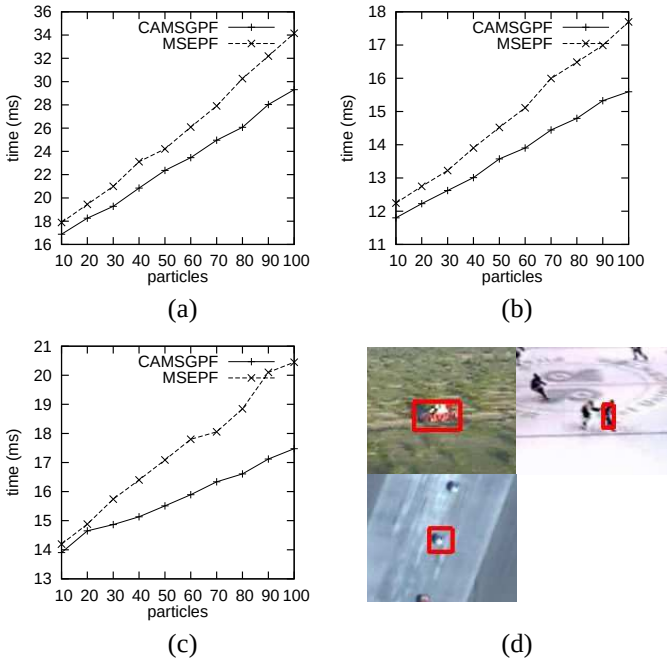

Fig. 5. Comparison of time consumption versus particle number between CAMSGPF and MSEPF. (a) "redteam"; (b) "hockey"; (c) "egtest01"; (d) tracking samples corresponding to (a)-(c)

than MSEPF. The curves in Fig. 5 do not pass through the origin, due to some fixed tasks in each frame (video file reading, color space conversion and displaying). So we deduct the fixed time for both of the two algorithms and compare the average time consumption per particle for each sequence, i.e., the slopes of the lines in Fig. 5, in the left columns of Table III. We can see it takes about 20%~50% less time for CAMSGPF to process a particle than MSEFP. In the right columns of Table III, the average Bhattacharyya distance between the tracked area and the target template is shown as an indicator of the tracking accuracy. All the 3 tests confirm that the influence due to simplification in CAMSGPF on the tracking accuracy is trivial with respect to the time saved.

IV. Conclusions

A novel tracking algorithm, CAMSGPF, has been proposed by exploring the interaction between particle filtering and CamShift. With the aids of CamShift, the particles are guided to more possible modes of observation, so that the sampling efficiency is improved greatly. At the same time, CamShift is conducted under the supervision of particle filtering. In this way, the scale adaptation of CamShift becomes functional in more universal situations; and CamShift can be applied on particles in a more economic way without much sacrifice in performance. The experimental results demonstrate that CAMSGPF outperforms CamShift and MSEPF in both tracking robustness and efficiency.

Acknowledgment

This work was supported by National Natural Science Foundation of China under Grant No.60502034, Shanghai

TABLE III

COMPARISON OF EFFICIENCY AND ACCURACY BETWEEN MSEPF AND CAMSGPF

Sequence	Time/Particle(ms)		Time Saved	Average distance*		Distance difference
	MSEPF	CAMSGPF		MSEPF	CAMSGPF	
Redteam	0.1807	0.1380	23.63%	0.1150	0.1140	-0.87%
Hockey	0.0618	0.0432	30.03%	0.0903	0.0950	5.13%
Egtest01	0.0691	0.0377	45.49%	0.1213	0.1311	8.14%

* a smaller distance indicates a better match between two objects.

Rising-Star Program under Grant No. 05QMX1435, Hi-Tech Research and Development Program of China (863) under Grant No. 2006AA01Z124, and Shanghai Postdoctoral Foundation under Grant No. 06R214138.

REFERENCES

[1] C. Shan, Y. Wei, T. Tan and F. Ojardias, "Real time hand tracking by combining particle filtering and mean shift," Sixth IEEE International Conference on Automatic Face and Gesture Recognition 2004.

[2] J.M. Hammersley and K.M. Morton, "Poor man's Monte Carlo," 1954. Journal of the Royal Statistical Society B, 16, 23-38

[3] M. Isard and A. Blake, "Condensation-conditional density propagation for visual tracking," International Journal on Computer Vision, 29(1), pp. 5-28, 1998.

[4] P. Perez, C. Hue, J. Vermaak and M. Gangnet, "Color-Based Probabilistic Tracking," ECCV, 2002, pp. 661-675.

[5] D. Comaniciu, V. Ramesh and P. Meer, "Real-time tracking of non-rigid objects using mean shift," IEEE Conference on Computer Vision and Pattern Recognition, pages II: 142-149, Hilton Head, SC, June 2000.

[6] G.R. Bradski, "Computer vision face tracking as a component of a perceptual user interface," the Workshop on Applications of Computer Vision, pages 214-219, Princeton, NJ, Oct. 1998.

[7] N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to nonlinear and non-Gaussian Bayesian state estimation," Proceeding of IEE, F, vol. 140, pp. 107-113, 1993.

[8] Y. Cai, N. de Freitas, J. Little, "Robust Visual Tracking for Multiple Targets," ECCV, 2006, vol. 4, pp. 107-118