# Fast Multiple Object Tracking via a Hierarchical Particle Filter

Changjiang Yang, Ramani Duraiswami and Larry Davis
Department of Computer Science, Perceptual Interfaces and Reality Laboratory
University of Maryland, College Park, MD 20742, USA
{yangcj,ramani,lsd}@umiacs.umd.edu

## Abstract

*A very efficient and robust visual object tracking algorithm based on the particle filter is presented. The method characterizes the tracked objects using color and edge orientation histogram features. While the use of more features and samples can improve the robustness, the computational load required by the particle filter increases. To accelerate the algorithm while retaining robustness we adopt several enhancements in the algorithm. The first is the use of integral images [34] for efficiently computing the color features and edge orientation histograms, which allows a large amount of particles and a better description of the targets. Next, the observation likelihood based on multiple features is computed in a coarse-to-fine manner, which allows the computation to quickly focus on the more promising regions. Quasi-random sampling of the particles allows the filter to achieve a higher convergence rate. The resulting tracking algorithm maintains multiple hypotheses and offers robustness against clutter or short period occlusions. Experimental results demonstrate the efficiency and effectiveness of the algorithm for single and multiple object tracking.*

## 1 Introduction

Object tracking is an important task in many computer vision applications including surveillance, gesture recognition, smart rooms, vehicle tracking, augmented reality, video compression, and medical imaging, etc. The tracking of real-world objects is a challenging problem due to the presence of noise, occlusion, clutter and dynamic changes in the scene other than the motion of objects of interest. A variety of tracking algorithms have been proposed and implemented to overcome these difficulties; they can be roughly classified into two categories: deterministic methods and stochastic methods.

Deterministic methods typically track by performing an iterative search for the local maxima of a similarity cost function between the template image and the current image. The cost function widely used is the sum of squared differences (SSD) between the template and the current image such as in [24, 16, 2]. More robust similarity measures have been applied and the mean-shift algorithm or other optimization techniques have been utilized to find the optimal solution [3, 5, 8, 11, 7]. Model-based tracking algorithms incorporate *a priori* information about the objects to develop representations such as skin complexion, body blobs, kinematic skeleton, silhouettes or layer information [38, 36, 5, 33, 32, 6]. Appearance-based approaches apply recognition algorithms to learn the objects either in some basis such as the eigenspace formed from observations or in kernel space [4, 1, 35].

On the other hand, the stochastic methods use the state space to model the underlying dynamics of the tracking system. In a linear-Gaussian model with linear measurement, there is always only one mode in the posterior probability density function (pdf), the Kalman filter propagates and updates the mean and covariance of the distribution. For nonlinear or non-Gaussian problems, it is impossible to evaluate the distributions analytically and many algorithms have been proposed to approximate them. The particle filter, also known as sequential Monte Carlo [10], is the most popular approach which recursively constructs the posterior pdf of the state space using Monte Carlo integration. It has been developed in the computer vision community and applied to tracking problem and is also known as the Condensation algorithm [17].

The particle filter based tracking algorithms usually use contours, color features, or appearance models [17, 18, 25, 37, 20]. The color histogram is robust against noise and partial occlusion, but suffers from illumination changes, or the presence of the confusing colors in the background. Most of all, it ignores the spatial layout information. The computation is expensive if the tracked region and the number of samples are large. The contour-based methods are invariant against the illumination variation but computationally expensive which restricts the number of samples (particles). Unfortunately when the dimensionality of the state space increases, the number of samples required for the sampling increases exponentially.

To resolve these problems, we adopt the Harr-like rectangle features introduced by Viola and Jones [34] for object detection, and the edge orientation histogram. The rectangle features can be efficiently evaluated by a few table lookup operations from the integral image as shown in [34]. The Harr-like rectangle features yield satisfactory results unless there are confusing pieces of the background with color similar to the foreground, or if there are significant illumination changes.

The edge or contour features are more robust to illumination variation or presence of confusing background colors, but are sensitive to clutter. One natural way to improve both the color and edge features is to combine them as in [3, 18, 37]. However, these earlier methods dealing with the edges are either time-consuming or restricted to simple shape models. To provide a general way to treat edge features instead we manipulate them using the edge orientation histogram (EOH) [15]. There are two reasons for this: one is that they can be efficiently computed in the same way as the rectangle features using the integral image; the other is that they are robust to scene illumination changes and provides more information than a simple contour model. Since both features can be evaluated efficiently, we can generate many more samples to alleviate the curse of the dimensionality and improve robustness.

While evaluating the probability that samples in the target image appear from the distribution of the original object, we will find that for most samples the probability is extremely low, and they can be pruned immediately by a bootstrap step in the particle filter. The few surviving samples are subjected to a more careful scrutiny in the following stages. In these following stages, more complicated and more discriminative features can be used to remove ambiguities in the first stage. This coarse-to-fine cascade idea, where the first stages reject most incorrect matches, while retaining all correct and a few incorrect ones, has been successfully used in target recognition and face detection [39, 13, 22, 34, 30]. As will be seen it allows for significant speed-up.

While employing extensively the integral image and the coarse-to-fine cascade scheme, our method is totally different from the object detection algorithms or face detection such as [39, 13, 22, 34, 30]. Temporal correlation in the sequences is ignored in these detection algorithms. If they are naively applied to object tracking, the false alarms or false negatives are too high for continuous tracking applications.

A further speed-up is made possible by improving the convergence of the Monte Carlo integration in the particle filter. We do so by using a quasi-random generator to generate the sample points [28]. To improve the running of the tracking algorithm we also employ the Streaming SIMD Extensions (SSE) and SSE2 instructions in the Pentium 4 processor. These are especially useful to improve the performance of computing the integral images. All these techniques allow our algorithm to comfortably run in real-time on a PC desktop.

## 2 Particle Filter

The particle filter is a Bayesian sequential importance sampling technique, which recursively approximates the posterior distribution using a finite set of weighted samples. It consists of essentially two steps: prediction and update. Given all available observations $\mathbf{z}_{1:t-1} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{t-1}\}$ up to time $t-1$, the prediction stage uses the probabilistic system transition model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ to predict the posterior at time $t$ as

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1}. \quad (1)$$

At time $t$, the observation $\mathbf{z}_t$ is available, the state can be updated using Bayes' rule

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})}, \quad (2)$$

where $p(\mathbf{z}_t|\mathbf{x}_t)$ is described by the observation equation.

In the particle filter, the posterior $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ is approximated by a finite set of $N$ samples $\{\mathbf{x}_t^i\}_{i=1,\ldots,N}$ with importance weights $w_t^i$. The candidate samples $\tilde{\mathbf{x}}_t^i$ are drawn from an importance distribution $q(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$ and the weight of the samples are

$$w_t^i = w_{t-1}^i \frac{p(\mathbf{z}_t|\tilde{\mathbf{x}}_t^i)p(\tilde{\mathbf{x}}_t^i|\mathbf{x}_{t-1}^i)}{q(\tilde{\mathbf{x}}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})}. \quad (3)$$

The samples are resampled to generate an unweighted particle set according to their importance weights to avoid degeneracy. In the case of the bootstrap filter [10], $q(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and the weights become the observation likelihood $p(\mathbf{z}_t|\mathbf{x}_t)$.

## 3 Observation Models

The observation model is is used to measure the observation likelihood of the samples, and is an important issue for object tracking. Many observation models have been built for particle filtering tracking. In [17], a contour based appearance template is chosen to model the target. The tracker based on a contour template gives an accurate description of the targets but performs poorly in clutter and is generally time-consuming. The initialization of the system is relatively difficult and tedious. In contrast, color-based trackers are faster and more robust, where the color histogram is typically used to model the targets to combat the partial occlusion, and non-rigidity [27, 37, 3, 8]. The drawback of the color histogram is that spatial layout is ignored, and the trackers based on it are easily confused by a background with similar colors.

The combination of the two features provides better performance at the price of trading speed for robustness. The color features provide robust and discriminative description of the objects using the body information. The edge features provide a discriminative description of the objects using the boundary information. Such a combination significantly improves the robustness and discriminative power of the features at the price of high computational load and slow tracking speed.

In order to resolve the contradiction between the robustness and the tracking speed, we use the simple rectangle features which have been used by Viola and Jones in object detection [34] and can be used in the bootstrap step as discussed below. These simple features can be efficiently evaluated by several table lookup operations on the integral image. The features are obtained on the grayscale images and can be easily extended to color images.

The edge information is represented using the edge orientation histogram (EOH). The EOH has been widely used for gesture recognition [15], pose estimation [31], distinctive image feature extraction [23], and object detection [21]. The reasons for using the EOH is that it is invariant to scene illumination changes, is discriminative against a background with confusing colors, and is simple and fast to compute.

### 3.1 Color Rectangle Features

The rectangle features were introduced by Viola and Jones for real-time object detection [34]. In their method, the grayscale image was converted to integral image format (an image in which at each pixel the value is the sum of all pixels above and to the left of the current position). The sum of the pixels within any rectangle can then be computed in four table lookup operations on the integral image. The color images can be treated as multi-channel intensity images to generate multi-channel integral images. The computation can be speeded up by using the SSE/SSE2 instructions that are available on the Pentium 4 CPU, where 128-bit instructions can be used to manipulate four 32-bit integers simultaneously.

To model the target using color information, we pick $n$ rectangular regions $R_1, \ldots, R_n$ within the object to be tracked. Each rectangle $R_i$ is represented by the mean $(r, g, b)$ color of the pixels within region $R_i$ (other color spaces can be considered similarly)

$$(r_i, g_i, b_i) = \sum_{(x,y)\in R_i} (r(x,y), g(x,y), b(x,y)) / A_i, \quad (4)$$

where $A_i$ is the number of pixels within $R_i$. The mean color vector $(r_i^\star, g_i^\star, b_i^\star)$ of each region $R_i$ can be computed during initialization. The reason we choose this color representation instead of the popular color histogram is that it encodes the spatial layout of the targets and offers robustness against noise. Furthermore, most of the targets consist of several homogenous sub-regions which makes the color histogram an inefficient representation.

Such a color representation of the targets has been used in [12] for head tracking, where a hypothesize-and-test procedure is used to find a match between frames. In [12], for real time performance they can only consider a relatively small number of hypotheses, since there was no efficient way to evaluate the rectangle features. This leads to decreased robustness. Here we have such an efficient way and can consider more hypotheses.

If we denote $\mathbf{k}^\star = \{(r_i^\star, g_i^\star, b_i^\star)\}_{i=1,\ldots,n}$ as the reference color model and $\mathbf{k}(\mathbf{x}_t)$ as the candidate color model, the similarity between $\mathbf{k}^\star$ and $\mathbf{k}(\mathbf{x}_t)$ can be measured by the Euclidean distance between them

$$\rho(\mathbf{k}^\star, \mathbf{k}(\mathbf{x}_t)) = \left[\sum_{i=1}^{n} (r_i^\star - r_i)^2 + (g_i^\star - g_i)^2 + (b_i^\star - b_i)^2\right]^{\frac{1}{2}}. \quad (5)$$

The likelihood distribution is given by

$$p(\mathbf{z}_t|\mathbf{x}_t) \propto e^{-\rho^2(\mathbf{k}^\star, \mathbf{k}(\mathbf{x}_t))/\sigma^2}, \quad (6)$$

where $\sigma = 10$ in our experiments.

The number of the rectangles within the object can be as small as two in the first stage which we will discuss later in this paper. There are two reasons for such a setup: one is efficiency, the other is that we want to keep more candidates in the first stage for robustness and prune those majority negative samples as soon as possible. This strategy has been proven successful in object detection [34], and we observe the same for tracking.

### 3.2 Edge Orientation Histogram

The color features are reliable for most tracking tasks, even when there is occlusion or overlap. However, it may perform poorly if the background presents confusing colors. Many other feature types have been proposed for combination with color. Isard and Blake [18] have shown that the combination of color with a contour model gives faster and more robust tracking. The color histogram and an ellipse shape model are combined for object tracking [3, 37]. In this paper, we use the edge orientation histogram for the purpose of simplicity, efficiency and generalization. It has been widely used in a variety of vision applications [15, 31, 23, 21, 9].

To detect edges, we first convert color images to grayscale intensity images. Edges are detected using the horizontal and vertical Sobel operators: $K_x$ and $K_y$ [14]:

$$G_x(x,y) = K_x * I(x,y), \quad G_y(x,y) = K_y * I(x,y). \quad (7)$$

The strength and the orientation of the edges are

$$S(x,y) = \sqrt{G_x^2(x,y) + G_y^2(x,y)}, \quad (8)$$

$$\theta = \arctan\left(G_y(x,y)/G_x(x,y)\right). \quad (9)$$

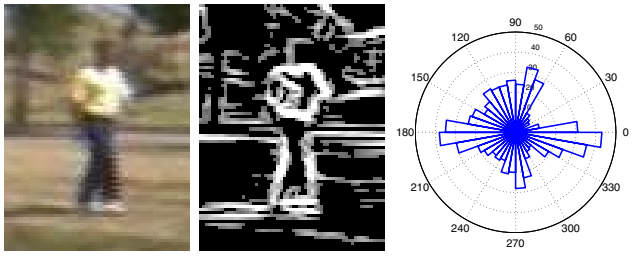


**Figure 1:** Edge orientation histogram. (*Left*) Example image. (*Center*) Edge strength image. (*Right*) Polar plot of edge orientation histogram.

We also apply a threshold $T$ to $S(x, y)$ to remove noise ($T$ is set to 100 in our experiments). The edges are counted into $K$ bins with their strengths $S(x, y)$. Figure 1 shows an example of the global edge orientation histogram of the walker in the image.

The edge orientation histogram can be built without explicitly computing the angles of the edges. Instead we use the normalized horizontal and vertical strengths $g_x = G_x/S$ and $g_y = G_y/S$ to index the edges into $K$ bins. The algorithm gives satisfactory results even when $K$ is as small as 4.

The edge orientation histogram within a rectangle region can be efficiently computed by treating it as $K$ separated channels and accumulating $K$ integral images. The $i$-th bin value within a rectangle is the sum computed by four table lookup operations on $i$-th integral image.

The similarity between the template and the current image is computed with the Euclidean distance between the two global edge orientation histograms as in [23]. The observation likelihood is calculated similarly using an expression such as (6).

### 3.3 Cascade of Features

The combination of the color information and edge orientation histogram achieves excellent performance in term of speed and accuracy. Typically the scores of about 90% of samples are almost zero (see the example in figure 2). The remained samples are subjected to a more careful examination. This cascade-like method has been widely used in many applications such as [39, 13, 22, 34, 30]. In particular, Viola and Jones [34] construct a cascade of classifiers which yields a very fast face detector. The use of cascade is effective because a majority of the sub-regions are negative. The cascade tries to reject as many as possible at the earlier stage, to concentrate the algorithm effort on the positive sub-regions.

Since the probability of most samples is almost zero, we can cut them off from the second stage evaluation. More sophisticated and more discriminative features and observation models are used in the second stage. In our implementation, we use the *two-rectangle*, *three-rectangle* and *four-rectangle* features in [34] as shown in Figure 3. Those features were used by the classifiers in the face detector [34]. We use them in the second stage because they are more discriminative at the same time more sensitive to the presence of edges, change in illumination or noise. The likelihood distribution of the subset of samples can be calculated similarly as in the first stage, then multiplied by the likelihood of the first stage. The probability of the remaining samples is scaled down by the smallest likelihood from the second stage. The samples with significant low probability will be neglected in the following stages.

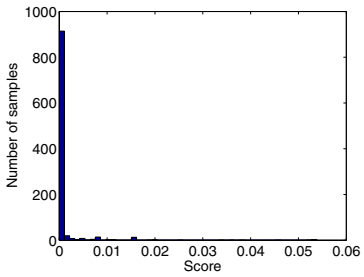


**Figure 2:** The histogram of scores of 1000 samples from an example sequence. Most samples have a vanishingly small score.

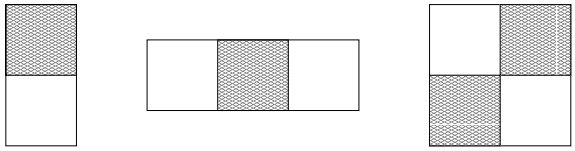

**Figure 3:** Rectangle features used in the paper, which are the remainders of the sums of the white rectangles from the gray ones. (*Left*) two-rectangle feature. (*Center*) three-rectangle feature. (*Right*) four-rectangle feature.

The features in the third stage is the edge orientation histogram which is similar to the one in the Scale Invariant Feature Transform (SIFT) descriptor [23]. The SIFT descriptor has achieved great success as a method for reliable image matching. The region is divided into $m \times n$ subregions and local histograms with 4 orientation bins are constructed in each by using the orientation integral images. The local edge orientation histograms are stacked into a multidimensional feature vector. The similarity between the template and the current image is computed using the Euclidean distance between the two multidimensional feature vectors. The observation likelihood is calculated using the expression (6). Each likelihood is multiplied by the previous one and the others are multiplied by the smallest likelihood in the third stage. In principle, more complicated features or representations can be used to construct the further levels of the cascade, but we restrict ourselves to these here.

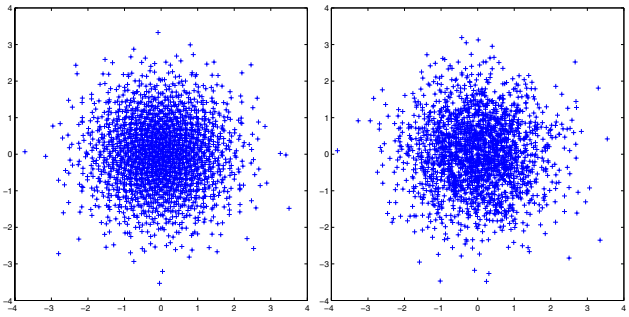

**Figure 4:** 2048 points of two-dimensional Gaussian random sequences generated by (*Left*) quasi-random sequence generator and (*Right*) pseudo-random sequence generator. The quasi-random sequence is more symmetric and has less discrepancy.

# 4 Particle Filter Tracking

The proposed particle filter tracker consists of an initialization of the template model and a sequential Monte Carlo implementation of a Bayesian filtering for the stochastic tracking system. In each iteration, the particle filter tracking algorithm consists of two steps: prediction and update.

The state of the particle filter is defined as $\mathbf{x} = (x, y, s_x, s_y)$, where $x, y$ indicate the location of the target, $s_x, s_y$ the scales in the $x$ and $y$ directions. In the prediction stage, the samples in the state space are propagated through a dynamic model. The dynamics usually is an auto-regressive process (AR). We use a first-order AR model for fair comparison and simplicity:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{v}_{t-1}, \tag{10}$$

where $\mathbf{v}_{t-1}$ is a multivariate Gaussian random variable. To draw samples from the normal distribution, we use the quasi-random sequence generator which converges in rate of $(\ln N)^d/N$ in $d$-dimensional state space instead of $N^{-1/2}$ using the pseudo-random sequence generator [28]. Figure 4 shows the random dots generated by the two random sequence generators. Clearly, the quasi-random sequence is more symmetric and samples space with less discrepancy.

The update stage applies the observation models to estimate the observation likelihood for each samples, i.e., the weights of samples in the case of the bootstrap filter. Since the probability of most samples is negligible, a bootstrap resampling is necessary to avoid the degeneracy. We apply the bootstrap scheme in [29] which uses order statistics and has the computational complexity $O(N)$.

# 5 Experiments

The proposed particle filter based tracker has been implemented in C and tested on a 1.4GHz Pentium4 PC with 512MB memory. It has been applied to a variety of tracking scenarios and tasks, including single and multiple object tracking.

## 5.1 Single Object Tracking

In the first experiment, a single person in a video sequence is walking in an office environment under severe illumination condition. The image size of the sequence is $352 \times 240$. The top row in Figure 5 shows that the tracker follows the target consistently and robustly despite the large illumination variations. The average tracking time per frame is about $0.015s$ with $1000$ samples. The bottom row in Figure 5 shows the tracking results of the color histogram based tracker in [27] with the same system dynamics, same initial template and $100$ samples. The tracking rate is about $50fps$. The tracker totally loses the target after several frames. With $1000$ samples, the tracking speed for the original algorithm will be very low (about $5fps$).

In the second experiment, the sequence is captured from outdoor environment with cluttered background and large changes in body size and shape. The results are shown in Figure 6. The image size is $720 \times 480$ and the number of particles is $1000$. The average tracking rate is about $30fps$. The great benefit of the proposed algorithm over other particle filter based tracking methods is shown in the multiple object tracking.

## 5.2 Multiple Object Tracking

In the third experiment, we use the proposed algorithm to simultaneously track multiple objects. Each object is associated with a individual template and $1000$ particles. Each object moves independently. The proposed algorithm successfully tracked all objects through all frames. The image size is $352 \times 288$. Examples of the tracking results are shown in Figure 7. The tracking time with respect to the frame index is shown in the left panel of Figure 8. We also measured the average tracking time for single object with respect to the number of particles which is shown in the right panel of Figure 8. The same procedures and configurations are applied to the color histogram based tracker and the corresponding tracking time is shown in Figure 8. It indicates that the increase of the time for the proposed algorithm with respect to the number of particles is much slower than for the color histogram based method. Most of the time in our method is spent on building the integral images. Once the integral images are built, the evaluation of the observation likelihood by the proposed method is independent of the size of regions and is very efficient. In contrast, for the color histogram based method or other methods, the bottleneck is the building of the histograms whose complexity is proportional to the number of particles and the size of the regions.

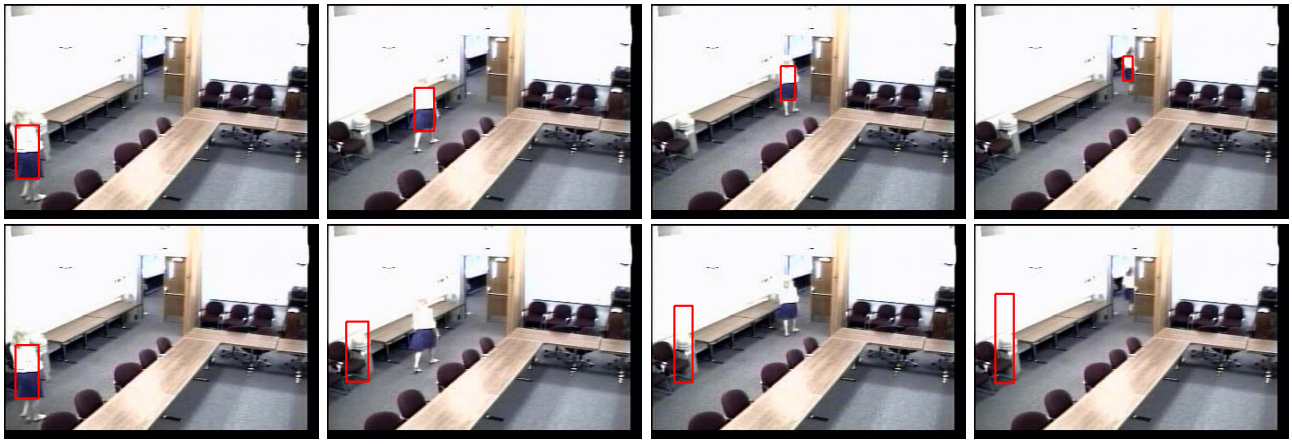

**Figure 5:** Particle filter based tracking in an office environment under severe illumination condition. (*Top*) The results of the proposed tracker. (*Bottom*) The results of the color histogram based tracker.

**Figure 6:** The results of the proposed particle filter based tracking for an outdoor sequence.

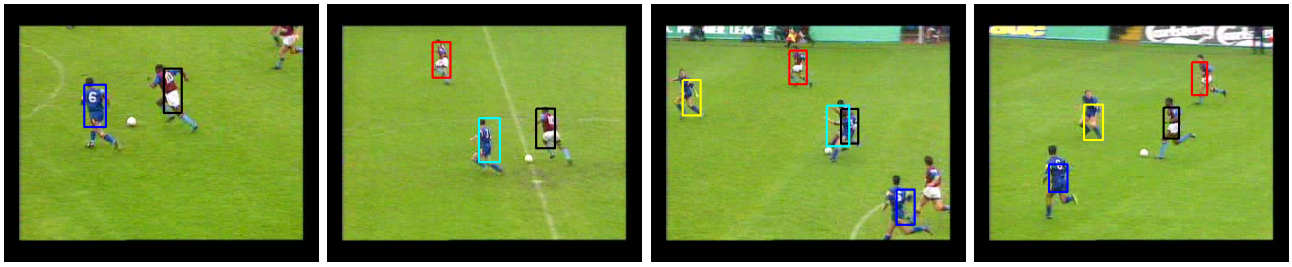

**Figure 7:** Results of the proposed particle filter based multiple object tracking for the football game sequence.

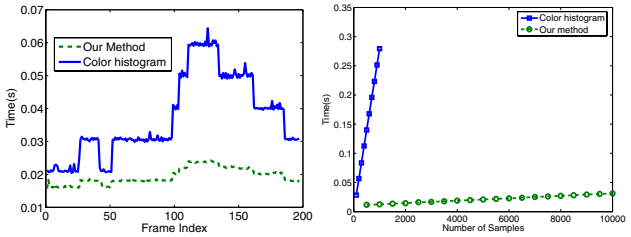


**Figure 8:** (*Left*) Tracking time with respect to the frame index. (*Right*) Tracking time with respect to the number of samples.

## 6 Conclusions

In this paper, we proposed an efficient and robust particle filter based object tracking algorithm. The particle filter maintains multiple hypotheses about the state of the tracked objects by representing the state space by a set of weighted samples. In general, the more samples and richer target representation, the better chances the tracking algorithms succeed in cluttered and noisy environments. However, they can be very inefficient when the popular color histograms, shapes, contours or a combination of these are used to evaluate the observation likelihood. We use the rectangle features and edge orientation histogram to evaluate the observation likelihood. They can be efficiently computed with integral images. To increase the discriminative capacity while speeding up the evaluation, we adopt a cascaded scheme which results in highly discriminative observation likelihood. The Monte Carlo integration in the particle filter converges at rate $O(N^{-1/2})$. We use the quasi-random sampling to improve the convergence rate to $O((\log N)^d/N)$ as in [28].

The above improvements make the tracking algorithm very efficient and robust against clutter, illumination changes and short period time occlusions. From the experiments, we find most of computations are spent on building the integral images. But once they are built, the evaluation of the observation likelihood can be done using several table lookup operations. So we can generate much more samples to represent the state space more accurately, which make it very suitable for multiple object tracking as in [26]. The probabilistic exclusion principle proposed by MacCormick and Blake [25] is very useful for tracking multiple objects, which employs the partition sampling to reduce the high computational cost from fully coupled systems. A combination of MacCormick and Blake's technique and ours will make the multiple object tracking more efficient and more robust. The object detection algorithm [34] also can be naturally integrated with our method. Currently the orientation of the targets is fixed during the tracking. The possible solution is to assign an orientation to the tracked objects using the edge orientation histograms as in [23]. In this paper we have not taken the updating of the image model into consideration. A good starting point will be work of Jepson et al [19].

## Acknowledgements

We would like to thank Sarnoff for providing the indoor sequence. We gratefully acknowledge the support of NSF grant IIS0205271 and DOD grant 2004H840200000.

## References

[1] S. Avidan. Support vector tracking. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, volume I, pages 184–191, Kauai, HI, 2001.

[2] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *Int'l Journal of Computer Vision*, 56(3):221–255, Feb. 2004.

[3] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 232–237, Santa Barbara, CA, 1998.

[4] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Int'l Journal of Computer Vision*, 26(1):63–84, 1998.

[5] G. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 2(Q2), 1998.

[6] G. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, volume I, pages 77–84, Madison, WI, 2003.

[7] R. Collins. Mean-shift blob tracking through scale space. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, volume II, pages 234–240, 2003.

[8] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–577, May 2003.

[9] J. W. Davis. Hierarchical motion history images for recognizing human motion. In *IEEE Workshop on Detection and Recognition of Events in Video*, pages 39–46, Vancouver, Canada, July 2001.

[10] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[11] A. Elgammal, R. Duraiswami, and L. Davis. Probabilistic tracking in joint feature-spatial spaces. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, volume I, pages 781–788, Madison, WI, 2003.

[12] P. Fieguth and D. Terzopoulos. Color based tracking of heads and other mobile objects at video frame rates. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 21–27, Puerto Rico, 1997.

[13] F. Fleuret and D. Geman. Coarse-to-fine face detection. *Int'l Journal of Computer Vision*, 41(1/2):85–107, 2001.

[14] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[15] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *Proc. Int'l Workshop on Automatic Face and Gesture Recognition*, pages 296–301, Zurich, Switzerland, June 1995.

[16] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(10):1025–1039, 1998.

[17] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *Int'l Journal of Computer Vision*, 29(1):5–28, Aug. 1998.

[18] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proc. European Conf. Computer Vision*, volume 1, pages 893–908, 1998.

[19] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(10):1296–1311, Oct. 2003.

[20] Z. Khan, T. Balch, and F. Dellaert. A rao-blackwellized particle filter for eigentracking. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, volume 2, pages 980–986, Washington DC, 2004.

[21] K. Levi and Y. Weiss. Learning object detection from a small number of examples: The importance of good features. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, volume 2, pages 53–60, Washington, DC, 2004.

[22] S. Z. Li, Z. Zhang, H.-Y. Shum, and H. Zhang. FloatBoost learning for classification. In *Advances in Neural Information Processing Systems*, 2002.

[23] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int'l Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[24] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674?–679, 1981.

[25] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Proc. Int'l Conf. Computer Vision*, pages 572–578, 1999.

[26] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *Proc. European Conf. Computer Vision*, volume 1, pages 28–39, 2004.

[27] P. Pérez, C.Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *Proc. European Conf. Computer Vision*, pages 661–675, 2002.

[28] V. Philomin, R. Duraiswami, and L. Davis. Quasi-random sampling for condensation. In *Proc. European Conf. Computer Vision*, volume 2, pages 134–149, 2000.

[29] M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[30] H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, volume 2, pages 29–36, Washington, DC, 2004.

[31] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proc. Int'l Conf. Computer Vision*, volume 2, pages 750–757, Nice, France, 2003.

[32] C. Sminchisescu and B. Triggs. Kinematic jump processes for monocular 3D human tracking. In *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, volume I, pages 69–76, Madison, WI, 2003.

[33] H. Tao, H. S. Sawhney, and R. Kumar. Object tracking with bayesian estimation of dynamic layer representations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(1):75–89, Jan. 2002.

[34] P. Viola and M. J. Jones. Robust real-time face detection. *Int'l Journal of Computer Vision*, 52(2):137–154, May 2004.

[35] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *Proc. Int'l Conf. Computer Vision*, pages 353–360, Nice, France, 2003.

[36] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):780–785, 1997.

[37] Y. Wu. Robust visual tracking by integrating multiple cues based on co-inference learning. *Int'l Journal of Computer Vision*, 58(1):55–71, June 2004.

[38] J. Yang and A. Waibel. A real-time face tracker. In *Proceedings of WACV*, pages 142–147, Sarasota, FL, 1996.

[39] M.-H. Yang, D. Roth, and N. Ahuja. A SNoW-based face detector. In *Advances in Neural Information Processing Systems*, pages 862–868, 1999.