A Generic Approach to Simultaneous Tracking and Verification in Video

Baoxin Li, *Member, IEEE,* and Rama Chellappa, *Fellow, IEEE*

***Abstract*—In this paper, a generic approach to simultaneous tracking and verification in video data is presented. The approach is based on posterior density estimation using sequential Monte Carlo methods. Visual tracking, which is in essence a temporal correspondence problem, is solved through probability density propagation, with the density being defined over a proper state space characterizing the object configuration. Verification is realized through hypothesis testing using the estimated posterior density. In its most basic form, verification can be performed as follows. Given a measurement vector Z and two hypotheses H_1 and H_0, we first estimate posterior probabilities $P(H_0|Z)$ and $P(H_1|Z)$, and then choose the one with the larger posterior probability as the true hypothesis.**

Several applications of the approach are illustrated by experiments devised to evaluate its performance. The idea is first tested on synthetic data, and then experiments with real video sequences are presented, illustrating vehicle tracking and verification, human (face) tracking and verification, facial feature tracking, and image sequence stabilization.

***Index Terms*—Importance sampling, Monte Carlo method, object verification, visual tracking.**

I. Introduction

PROGRAMMING a computer to recognize objects is a difficult problem, and has been a research topic for many years. Recently, there has been increasing interest in integrating the temporal information available in video for improved recognition performance. However, in most cases, the temporal information is exploited only in tracking an object, with less emphasis on using temporal information for recognition. In its most crude form, temporal information can be exploited through voting, i.e., recognition is done on each frame, and a vote is taken to give the final decision. While this may be helpful in some cases, much information is being left out in this crude approach, such as temporal coherence in the shape changes of an object in consecutive frames.

With video data, recognition often becomes a verification problem. That is to say, an algorithm is needed to answer the question: is this the object seen at a previous time? Or, is this the object I was asked to look for? And often, there are only a small number of candidates to verify against, with the candidates being templates obtained from earlier parts of the video. The problem can be illustrated by the examples of monitoring a vehicle entering and then leaving a parking lot, and a person entering and then leaving a bank. In these scenarios, tracking is needed first; then the algorithm needs to do incremental verification while maintaining track on the object. Obviously, in these applications, temporal information from video, if properly exploited, would facilitate recognition.

In this paper, a generic approach to simultaneous object tracking and verification is proposed. The approach is based on posterior probability density estimation through sequential Monte Carlo methods. Tracking, which is in essence a temporal correspondence problem, is formulated as a probability density propagation problem, with the density being defined over a proper state space characterizing the object configuration. With a novel reparametrization, many tracking applications involving different representations such as edge maps, intensity templates, and feature point sets are uniformly processed by the same algorithm. Examples will also show how the approach can be applied to tasks like *image sequence stabilization*. In addition to performing tracking, the algorithm also gives *verification* results as time proceeds; this is realized through hypothesis testing using the estimated posterior probability densities.

The paper is organized as follows. Section II shows how tracking can be formulated as a Bayesian inference problem, and also gives a brief introduction to sequential Monte Carlo methods. Section III describes our approach to simultaneous tracking and verification. Applications of the approach are illustrated by experiments in Section IV. Section V relates our approach to other work. We conclude in Section VI.

II. Theoretical Background

A. Tracking as a Bayesian Inference Problem

Tracking is the processing of measurements obtained from an object in order to maintain an estimate of its current state, which typically consists of kinematic components (position, velocity, etc.) and other components (signal strength, "feature" information, etc.). Let X_t denote the state to be estimated, and let $\mathbf{Z}_t$ be the measurement vector (observations up to t: $\{Z_0, Z_1, \ldots, Z_t\}$). The subscript t denotes a discrete time index. Both X_t and $\mathbf{Z}_t$ are in general random quantities. Let $p(X_t)$ be the distribution of X_t; we then have the joint distribution $p(X_t, \mathbf{Z}_t)$: $p(X_t, \mathbf{Z}_t) = p(X_t) \cdot p(\mathbf{Z}_t|X_t)$, where

Manuscript received May 1, 2001; revised February 6, 2002. This work was supported by the Advanced Sensors Consortium (ASC) sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0001. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Thiow Keng Tan.

B. Li is with Sharp Laboratories of America, Camas, WA 98683 USA (e-mail: bli@sharplabs.com).

R. Chellappa is with the Center for Automation Research, University of Maryland, College Park, MD 20742 USA (e-mail: rama@cfar.umd.edu).

Publisher Item Identifier S 1057-7149(02)04781-4.

$p(\mathbf{Z}_t|X_t)$ is the likelihood of X_t, having observed $\mathbf{Z}_t$. Using Bayes theorem, we have

$$p(X_t|\mathbf{Z}_t) = \frac{p(X_t)p(\mathbf{Z}_t|X_t)}{\int p(X_t)p(\mathbf{Z}_t|X_t)\,dX_t}$$

which is the *posterior* distribution of X_t, and is what Bayesian inference attempts to estimate. Assuming we have obtained $p(X_t|\mathbf{Z}_t)$, tracking is solved; knowing X_t, by definition we know everything about the current state of the object, including its location and other dynamics in the state vector. Thus tracking can be formulated as a Bayesian inference problem, with $p(X_t|\mathbf{Z}_t)$ as the objects of the inference. Note that in this formulation, the posterior $p(X_t|\mathbf{Z}_t)$ is a time-variant quantity; in a tracking problem, $p(X_t|\mathbf{Z}_t)$ at time t is evolved from $p(X_{t-1}|\mathbf{Z}_{t-1})$ at time $t-1$. In this sense, tracking is also a density propagation problem.

In reality, instead of obtaining the posterior density itself, a Bayesian inference task may focus on only estimating some properties of the density, such as moments, quantiles, highest posterior density regions, etc. All these quantities can be expressed in terms of posterior expectations of functions of X_t. The posterior expectation of a function $f(X_t)$ is

$$E[f(X_t)|\mathbf{Z}_t] = \frac{\int f(X_t)p(X_t)p(\mathbf{Z}_t|X_t)\,dX_t}{\int p(X_t)p(\mathbf{Z}_t|X_t)\,dX_t}.$$

The integration in this expression has until recently been the source of most of the practical difficulties in Bayesian inference, especially in high dimensions. In most applications, analytic evaluation of $E[f(X_t)|\mathbf{Z}_t]$ is impossible. Alternatives include numerical evaluation, for example, using classical quadrature. However, in high dimensions, this method is computationally prohibitive since too many samples will be required to yield reasonable accuracy.

An important alternative technique is Monte Carlo integration (e.g., [14], [21]), which, in general, is often suitable for high-dimensional integration. The basic Monte Carlo method approximates a definite integral by uniformly sampling from the domain of integration, and averaging the function values at the samples. There are two major limitations to the basic Monte Carlo approach: 1) the accuracy improves only linearly with the number of samples and 2) more samples are needed if the integrand has peaks in some small regions and is very small elsewhere. Common methods of handling these limitations (especially the second one) include importance sampling, rejection sampling, Gibbs sampling (which can be treated as a single-component Markov Chain Monte Carlo (MCMC) approach), etc. There also exist techniques for combining these methods for improved performance. One elegant approach is the sequential importance sampling method, which is discussed in the next subsection.

B. Sequential Monte Carlo Methods for Dynamic Systems

The state X_t and observation $\mathbf{Z}_t$ in a tracking problem are time-variant quantities of a time-variant system, for which the state space model is a popular way of analyzing. When the system is linear and the noise model is Gaussian, the use of the Kalman filter has been a common practice, and optimal results can be obtained with the Kalman filter. Extended Kalman filters and other approximations have been utilized to handle nonlinear and/or non-Gaussian systems (e.g., [1], [28]).

In recent years, Monte Carlo methods have been proposed for the analysis of nonlinear and/or non-Gaussian dynamic systems (e.g., [7], [22]). One class of Monte Carlo methods is the so-called *sequential importance sampling* (SIS) methods. In SIS methods, at time t, the dynamic density is approximated by a set of its samples, *with proper weights*. This is basically importance sampling [14] used in a sequential fashion. SIS has some of the good properties of both importance sampling and MCMC. On the one hand, the samples are properly weighted, and the significance of a sample is represented by its weight, which can be updated by incorporating current observations. On the other hand, by re-using the samples, SIS can keep track of a slowly varying density. Applications of SIS can be found, for example, in target position tracking [13], in the Bayesian missing data problem [23], and in contour tracking [20].

We now give a brief description of the SIS approach. Following [27], one first characterizes a probabilistic dynamic system as a sequence of evolving probability distributions $\pi_t(X_t)$, indexed by discrete time t, where X_t is the state variable at time t. This is more general than the state space model and can handle other problems such as the Bayesian missing data problem. In this setting, the posterior estimation problem in a state space model is a special case, with $\pi_t(X_t) = p(X_t|\mathbf{Z}_t)$. An SIS algorithm, which summarizes the aforementioned methods [13], [23], [20], is then designed as shown in the following algorithm (for a complete treatment, see [27]).

SIS Algorithm

Let $S_t = \{X_t^{(j)},\ j = 1, \ldots, m\}$ denote a set of random draws that are properly weighted by the set of weights $W_t = \{w_t^{(j)},\ j = 1, \ldots, m\}$ with respect to π_t. At each time step t,

step 1. Draw $X_{t+1} = x_{t+1}^{(j)}$ from $g_{t+1}(x_{t+1}|x_t^{(j)})$;

step 2. Compute

$$u_{t+1}^{(j)} = \frac{\pi_{t+1}\left(x_{t+1}^{(j)}\right)}{\pi_t\left(x_t^{(j)}\right) g_{t+1}\left(x_{t+1}^{(j)}|x_t^{(j)}\right)} \text{ and } w_{t+1}^{(j)} = u_{t+1}^{(j)} w_t^{(j)}.$$

Then $(x_{t+1}^{(j)},\ w_{t+1}^{(j)})$ is a properly weighted sample of π_{t+1}.

In this algorithm, $g(\cdot)$ is called the trial distribution or proposal distribution. Although any choice of $g(\cdot)$ will ultimately deliver samples from the desired distribution (subject to regularity conditions; see, for example, [32]), the rate of convergence will depend crucially on the relationship between $g(\cdot)$ and $\pi(\cdot)$. There are different proposals about the choice of $g(\cdot)$, such as in [23] and [22]. The latter is used in our work, which will be described in detail in Section III-D.

When a sequence of observations induced by a dynamic object parametrized by X is available, such as a video sequence of

a moving person, the posterior density estimation can be solved in a sequential way using the SIS algorithms. With a state-space model, the weight can also be deduced from the likelihood of the state $p(\mathbf{Z}_t|X_t^{(j)})$, which incorporates the current observation [22], [20].

III. Simultaneous Tracking and Verification via Posterior Estimation

With the SIS algorithm, tracking is immediately solved by setting the state X to some parametrization of the object that includes the location of the object. For example, let X be the two-dimensional (2-D) translation of a 2-D object. Using the SIS algorithm, one can get the density of X at each time t. If the expectation of X is used as the estimate of the true translation of the object, tracking is achieved after approximating the expectation using the estimated density.

Assuming that tracking has been done, we now consider verification. In Section III-A, we first put the verification problem into a probabilistic setting. Section III-B introduces a reparametrization which leads to a dynamic system of lower dimensionality. Section III-C gives our algorithm for simultaneous tracking and verification based on the SIS method. Section III-D discusses implementation issues of the algorithm.

A. Verification via Posterior Probability

Assume that there are C classes $\{\omega_1, \ldots, \omega_C\}$ (e.g., C different people). Let X be a parametrization of the object, which can be, for example, the intensity image of the object (viewed as a vector). In general, X is a random vector governed by the *a priori* density $p(X|\omega_i)$. Given an observation $\mathbf{Z}$, the Bayesian *maximum-a-posteriori-probability* rule chooses $\omega = \max_i\{P(\omega_i|\mathbf{Z})\}$ as the solution to the verification problem, where $P(\omega_i|\mathbf{Z})$ is the posterior probability of class ω_i given the observation $\mathbf{Z}$. $P(\omega_i|\mathbf{Z})$ can be computed through integrating the posterior density as

$$P(\omega_i|\mathbf{Z}) = \int_A p_i(X|\mathbf{Z})\, dX \tag{1}$$

where $p_i(X|\mathbf{Z})$ is the posterior density of class ω_i, with A being some proper region. Therefore, the verification problem can be solved best (in the Bayesian sense) if the posterior $p_i(X|\mathbf{Z})$ is first estimated. Let hypothesis H_i denote the event "class ω_i causes the observation $\mathbf{Z}$." We then use $P(\omega_i|\mathbf{Z})$ and $P(H_i|\mathbf{Z})$ interchangeably.

Unfortunately, when X is high-dimensional (for example, an image vector), posterior estimation through empirical approaches is not realistic, even with SIS methods. Given limited observation data, the estimates would be inaccurate at best, and meaningless at worst. If, however, the object can be characterized by a vector of low dimensionality, then the SIS method would be an effective tool for posterior estimation.

B. Reparametrization

Consider a rigid object subject to motion which can be modeled by a transformation f parametrized by a parameter vector θ. Let X_0 denote an original parametrization of the object. X_0 can be, for example, a face template (intensity image), a set of intensity discontinuity points (edge map), or the parametric contour of an object. Let $X = f(\theta, X_0)$ denote the transformation of X_0 into X. Under the small and continuous motion assumption, X is similar to X_0, meaning that θ has only a "small" difference from θ_0, with θ_0 being the parameter for the identity transform: $X_0 = f(\theta_0, X_0)$. Expanding $X = f(\theta, X_0)$ at θ_0 gives

$$\begin{aligned} X &= f(\theta, X_0) \\ &= f(\theta_0, X_0) + J_\theta(\theta_0)(\theta - \theta_0) + o(\cdot) \\ &\approx X_0 + J_\theta(\theta_0)(\theta - \theta_0) \end{aligned} \tag{2}$$

where $o(\cdot)$ denotes higher order terms, and $J_\theta(\cdot)$ is the Jacobian matrix with respect to θ. This expansion shows that the transformed object X can be viewed as the original X_0 plus a changing term caused by a $\Delta\theta \stackrel{\text{def}}{=} \theta - \theta_0$. From a practical point of view, only the difference is important; knowing this, temporal correspondence is solved. Given X_0, the vector $\Delta\theta$ is a good parametrization of all possible X under the small motion assumption. Thus, we propose to use $\Delta\theta$ as the state vector.

The Jacobian matrix $J_\theta(\theta_0)$ is easy to obtain for 2-D affine or simpler transformations. For example, let $X_0 = (x_0, y_0)$ be the location of one edge point. For a 2-D affine transformation $f(\theta, \cdot)$ defined by $f(\theta, \cdot) = \left(\begin{smallmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{smallmatrix}\right)(\cdot) + \left(\begin{smallmatrix} T_x \\ T_y \end{smallmatrix}\right)$, let $\theta \stackrel{\text{def}}{=} (a_{11}, a_{21}, a_{21}, a_{22}, T_x, T_y)^T$; then the Jacobian matrix is computed as

$$J_\theta(\theta_0) = \left|\frac{\partial X}{\partial \theta}\right|_{\theta_0} = \begin{pmatrix} x_0 & y_0 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_0 & y_0 & 0 & 1 \end{pmatrix}. \tag{3}$$

From now on, whenever X is used, it refers to $\Delta\theta$. That is to say, the dynamic system under consideration will be the one governing the evolution of $\Delta\theta$. It is worth pointing out that (2) is valid for general transformations other than 2-D affine or Euclidean similarity transformations. Also, since 2-D affine is linear in each component of θ, the higher-order terms in (2) are zero, implying that the parametrization with $\Delta\theta$ is accurate for a 2-D planar object even under large motion (this is also true for the 2-D translation and Euclidean similarity groups, see [25]).

C. The Algorithm

An algorithm for simultaneous tracking and verification is

Algorithm

Initialization: Rectify the templates onto the first frame of the sequence.

Tracking:

Initialization: draw N_s random samples from $\pi_0(X_0)$.

Updating: at time $t > 0$, invoke the SIS algorithm to obtain an updated set of samples for $\pi_t(X_t)$.

Verification:

At time $t > 0$, evaluate the mean value $E_\pi(X_t)$ of X_t;

Compute the posterior probability $P(H_i|\mathbf{Z}_t)$ according to (1), with A being a hypercube around

$E_\pi(X_t)$: $A = [E_\pi(X_t) - \Sigma,\ E_\pi(X_t) + \Sigma]$;

Choose as true the hypothesis H_j giving the maximum probability (or sum of probabilities up to t).

As shown previously, the algorithm is initialized by rectifying the templates to the first frame of the sequence. This step corresponds to the object detection process, which will not be considered in this paper. The rectification refers to registering the template to the scene, which is usually done approximately by the detection process. With this algorithm, this registration need not be very accurate, since in the second step, random samples will be drawn around the initial point, and will capture the real value with a probability depending on the variance σ_0 of the population and the sample size N_s. In theory, this probability goes to 1 with large σ_0 and N_s.When the state vector X is some parametrization of the underlying object shape or appearance, it is not hard to understand the rationale behind the verification step in the previous algorithm. It remains to be explained why we can still do verification using the algorithm when X is $\Delta\theta$. The argument is that given the observation history $\mathbf{Z}_t$, the true hypothesis should generate a density of higher peak and more concentrated shape than a false hypothesis would, since the transformed object should be "close" to the original one under small motion assumption. In an ideal (deterministic) situation the measurement of the object at time t can be related to its model (true hypothesis) by a unique X_t, while no transform can relate a false hypothesis (a model different from the right one) to the measurement without incurring a large matching error.

From a Bayesian point of view, recall that

$$p(X|\mathbf{Z}) = p(\mathbf{Z}|X)p(X)/p(\mathbf{Z})$$

where $p(\mathbf{Z})$ is the same for all the hypotheses and, thus, has no effect on verification, and $p(X)$ is the prior density of X. $p(X)$ is usually assumed to be unimodal, for example Gaussian. $p(\mathbf{Z}|X)$, which should be treated as a function of X when $\mathbf{Z}$ is given, reflects the likelihood of the event "the observation $\mathbf{Z}$ being incurred by X." Without considering occlusion, in general $p(\mathbf{Z}|X)$ should peak at the true X which causes the observation, and decrease when X deviates from the true value. It is reasonable to assume that $p(\mathbf{Z}|X)$ is unimodal when X reflects a small change with respect to some X_0. This is especially true in a verification problem where the templates are usually obtained from earlier frames of the sequence. The single subject assumption is automatically satisfied through tracking: only a local region needs to be considered, and within this region the assumption of a single subject is reasonable.

In summary, under the previous assumptions, $p(X|\mathbf{Z})$ should be unimodal with the peak at the true value of X. This is the basis for the verification step. Later, in our experiments, it will be observed that $p(\mathbf{Z}|X)$ is indeed shaped as predicted. It will also be shown that when the single object assumption is violated, the verification can fail temporarily (e.g., Fig. 9).

It must be pointed out that, when the system dynamics are known (such as a constant velocity model), another type of information is available to assist verification: the prediction error. In the algorithm, at time $t+1$, with the mean value of state X_t at time t known, prediction can be done using the system dynamics, and the prediction error gives an additional indication of whether the hypothesis is true. Ideally, a true hypothesis gives a smaller prediction error. In the application examples in this paper, we only assume that the system is governed by a first-order Gauss-Markov process (random walk). Thus we will not exploit the information from a prediction error perspective.

In the verification algorithm, a threshold Σ needs to be specified to define the integral region A. To avoid choosing an *ad hoc* Σ, one can compute the (e.g., 95%) confidence interval around the mean, and then test the hypotheses based on the length of the interval in each dimension. However, we will use the probability rather than the length of the confidence interval since the former is more intuitive.

Notice that in the previous algorithm, multiple hypotheses are kept during the tracking and verification process, meaning that several dynamic systems are maintained simultaneously. This is natural in applications—the actual scenario could be either of the following: the system has to identify a given object (template) from multiple objects in view (such as tracking a person who entered the bank a moment ago, based on a sequence containing multiple persons); or the system has to identify a single object as one of several possible candidates (templates) (such as classifying the person seen in a sequence as one of the candidates). Our verification experiments in the next section will concentrate on the latter case. Maintaining multiple systems increases the computational complexity. Fortunately, the systems can be processed in parallel.

D. Issues in Implementing the Algorithm

In introducing the algorithm in Section III-C, we did not specify details about implementing the SIS steps. We were intentionally obscure on that subject, because many techniques exist for improving the SIS algorithm; thus we wanted the algorithm to be generic without the constraint of a specific implementation. In this subsection, we discuss some implementation issues, especially the techniques used in this work.

Choice of the Proposal Distribution: As mentioned earlier, the choice of the proposal distribution $g(\cdot|\cdot)$ is important in practice. When the system dynamics are described by a state space model, with transition kernel (transition kernels are general versions of transition probabilities of a discrete state space model)

$$p_{t+1}(X_{t+1}|X_t,\ \mu)$$

where X is the state and μ the model parameters, a good choice of $g(\cdot|\cdot)$ is

$$g_{t+1}(X_{t+1}|X_t) = p_{t+1}(X_{t+1}|X_t,\ \mu).$$

With this choice of $g(\cdot|\cdot)$, u_{t+1} is proportional to the measurement likelihood of a particular sample $X_{t+1}^{(j)}$ which is given by

$$L_{t+1}^{(j)} = p_{t+1}\left(Z_{t+1}|X_{t+1}^{(j)}\right).$$

Thus the measurement is easily incorporated into the weight update of the samples. This choice of $g(\cdot|\cdot)$ has been used in [22], [13], [20], etc., and is what is used in our implementation of the SIS algorithm.

Fig. 1. Left: A car parametrized by intensity edges. This is what we used to synthesize the sequence in Fig. 2. Right: Another vehicle used as the alternative hypothesis.

Resampling: Other techniques also exist for improving the performance of the SIS algorithm, such as *resampling* (e.g., in [22]), *rejection sampling* (e.g., in [39]), *adaptive direction sampling* [12], *Rao–Blackwellization* [10], etc. However, most of these techniques do not always work well. For example, Rao–Blackwellization was intended for reducing the estimation variance, but it does not always work efficiently (see [11]). While it is possible to incorporate these techniques into the sampling step in the algorithm in Section III-C, our current implementation uses the resampling technique given in [22]. The resampling scheme generates, at time $t+1$, samples $X_{t+1}^{(j)}$ by resampling $X_t^{(\cdot)}$ with probabilities

$$P\left(X_{t+1}^{(j)} = X_t^{(i)} | Z_{t+1}\right) = L_t^{(i)} \Big/ \sum_k L_t^{(k)}$$

where the summation is over the set of samples. This is also partly inspired by the successful application of this simple resampling technique in contour tracking in [20] (where the algorithm is called CONDENSATION).

Measurement Likelihood: In applications, the measurement likelihood is obtained, for example, from the measurement equation of a state space model. Thus for different applications, one could have different expressions for the measurement likelihood. We will, however, use a simple measurement likelihood formula for all experiments in this work—a truncated Gaussian model. Of course, a Gaussian model may not be accurate for real applications, but it turns out that in the tracking problem, if the tracker is initialized well, a Gaussian model can be a reasonable approximation. This is especially helpful when it is difficult to specify the true measurement distribution analytically. Specifically, we define

$$L_t^{(j)} = \begin{cases} \dfrac{1}{\sqrt{2\pi}\sigma_0} \exp\left\{-\dfrac{\left(e_t^{(j)}\right)^2}{\sigma_0^2}\right\}, & \text{if } \left|e_t^{(j)}\right| < \delta \\ C, & \text{otherwise} \end{cases}$$

where $L_t^{(j)}$ and $e_t^{(j)}$ are the measurement likelihood and the prediction error, for sample j at time t, respectively, with δ being a threshold and C a constant. If we assume that L is time-homogeneous, then the subscript t of $L_t^{(j)}$ can be dropped.

Note that the prediction error $e_t^{(j)}$ is problem-dependent; thus we will not specify it until we are dealing with a specific application of the algorithm.

System Dynamics: As discussed earlier, the system dynamics, if known, can be incorporated into the SIS steps [for choosing $g(\cdot|\cdot)$], and the prediction error can be used to assist in hypothesis testing. However, in all the experiments in this paper, we only assume that the system is governed by a first order Gauss–Markov process (random walk). This enables the algorithm to work on unknown dynamics. To be specific, the conditional distribution density of state X_{t+1} given X_t is of the form (for simplicity, only the one-dimensional case is given)

$$p(x_{t+1}|x_t) \sim \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left\{-\frac{(x_{t+1}-x_t)^2}{\sigma_1^2}\right\}$$

where σ_1^2 is the variance, which controls, roughly speaking, how far X_{t+1} can be from X_t.

IV. Applications

In this section, we describe experiments that illustrate the applications of the proposed algorithm. Different input representations have been used in the examples. We first test the algorithm using synthetic sequences (based on edges), since synthetic data allow us to compare estimates with the true values. Next, applications to vehicle (based on edges) and human/face (based on intensity images) tracking and verification are demonstrated with real video data. A facial feature tracking application example is then presented, based on Gabor attributes on a set of grid points over a face. Finally, we give an interesting application of the algorithm to image sequence stabilization.

The sample size N_s is 200 for all the experiments in this paper. In general, using more samples would lead to more accurate approximation to the density, but also increase the computational complexity. We have found that $N_s = 200$ can provide fairly good results for the experiments reported in this section. The sequences used in the experiments have a frame resolution of 240×320 pixels unless specified otherwise.

A. Test on Synthetic Data

In the experiments using synthetic data, we assume that the sequence contains a car receding in the field of view, with its motion specified by a 2-D affine transformation. The car (in the form of its intensity edges) is shown in Fig. 1 (left).

While in real applications, the observations are obtained from real images through preprocessing techniques, e.g., [6], in this synthetic test, we simply transformed the template with an affine transformation at each time t and then added independent noise to each edge pixel. The noise is uniformly distributed on a region $[-\sigma, \sigma] \otimes [-\sigma, \sigma]$ centered at the current edge pixel, with σ controlling the noise magnitude. In addition, we discard each edge pixel from the observation with probability P_v, and we let each background pixel be falsely detected as an edge with probability P_f. An example of such an artificial "observation" sequence is shown in Fig. 2 (see also Movie 1[1]). Although this

[1]Each sequence referred to in this paper has a corresponding video clip at www.cfar.umd.edu/~baoxin/IP.html.

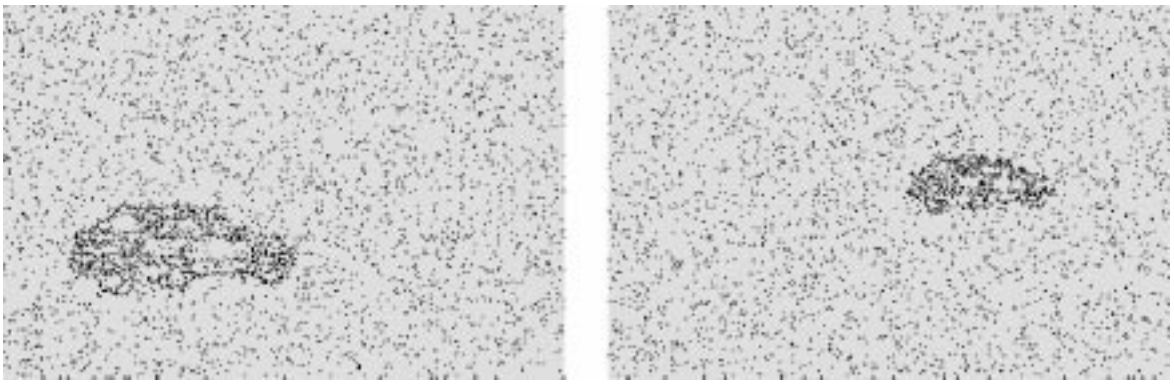

Fig. 2. Sample frames of a synthetic sequence (see text), simulating a car driving away (see also Movie 1). The original frame size is 240 × 320.

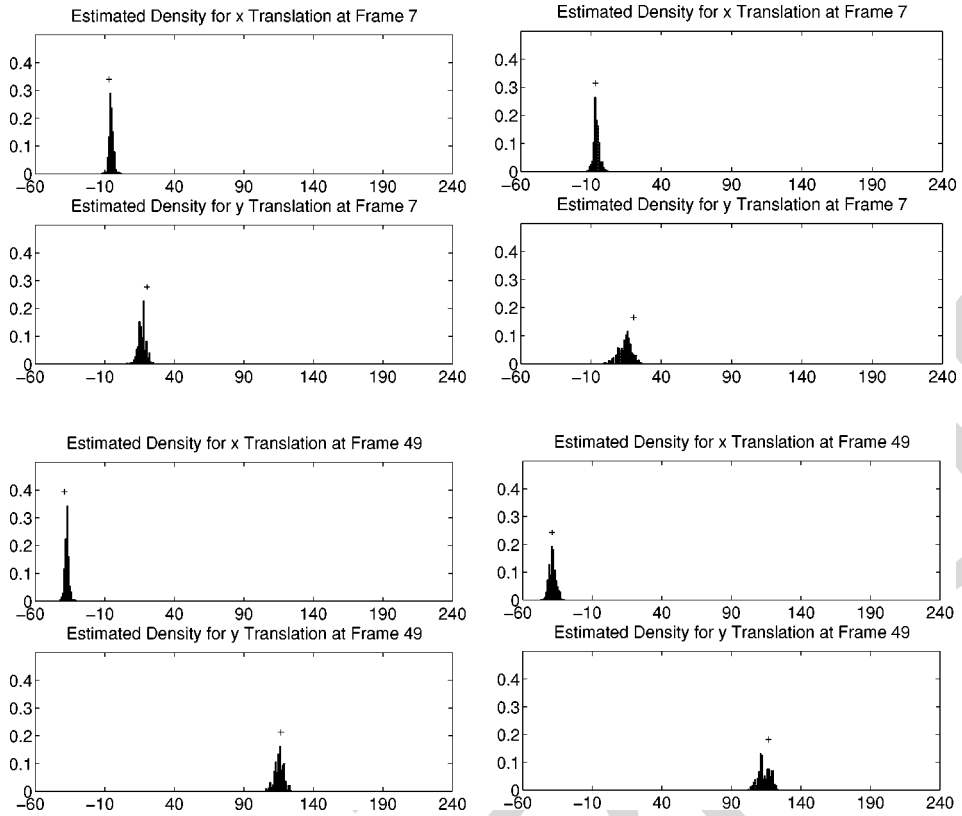

Fig. 3. The posterior densities for x and y translations at frames 7 and 49. The left column is obtained when the template is the true object appearing in the sequence; the right column is from a false candidate. The crosses indicate the ground truth at the corresponding frames.

may not model the real observation well, it is obvious that the artificial measurements are highly contaminated by non-Gaussian noise. The prediction error is computed by

$$e = \begin{cases} \dfrac{1}{n(M)} \displaystyle\sum_{a \in M} \\ \quad \inf_{b \in E_t} \|a - b\|, & \text{if } \inf_{b \in E_t} \|a - b\| < \delta_1 \\ C_1, & \text{otherwise} \end{cases}$$

where E_t is the edge point set from frame t, M the edge point set from the model image, and $n(M)$ the number of points in M. C_1 is a constant and δ_1 a threshold. $\|\cdot\|$ is the L_2 norm. This is in a sense like an averaged partial Hausdorff distance [17].

With X_0 being the parametrization of the template, we first estimate the posterior $p(X_t|\mathbf{Z}_t)$ from the synthetic sequences. The true and alternative hypotheses (both shown in Fig. 1) are then both tested against the sequence. Fig. 3 shows an example of the estimated posterior probability densities for x and y translation, with the ground truth marked by a cross, for the true and alternative (false) candidates, respectively. The resemblance of the two densities results from the fact that the two templates are very similar. Yet, a close look at the figure shows that the true template generates more concentrated densities with higher peaks.

To better show the effectiveness of the algorithm, we plot three of the estimated motion parameters [i.e., $E_\pi(X_t)$], together with the ground truth, in Fig. 4. This figure clearly shows

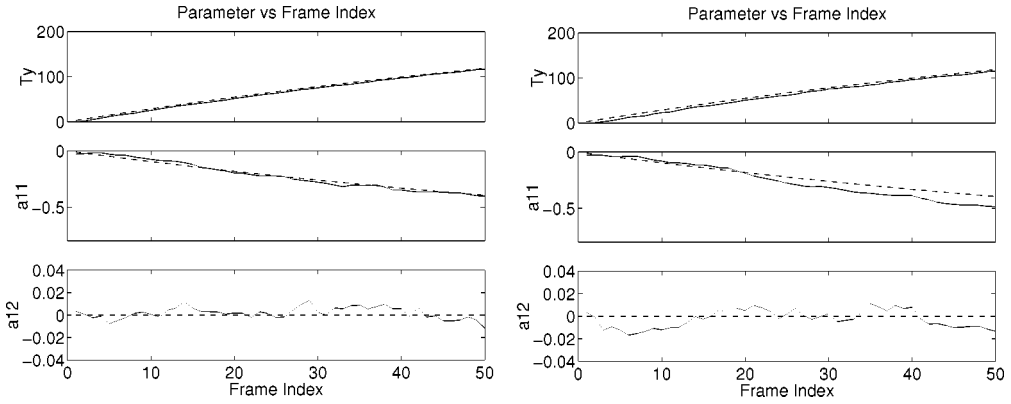

Fig. 4. True (dashed) and estimated (solid) parameters versus frame index. Top: obtained from the true hypothesis. Bottom: obtained from the false hypothesis. Note that the parameters should be interpreted as $\theta - \theta_0$ (see text).

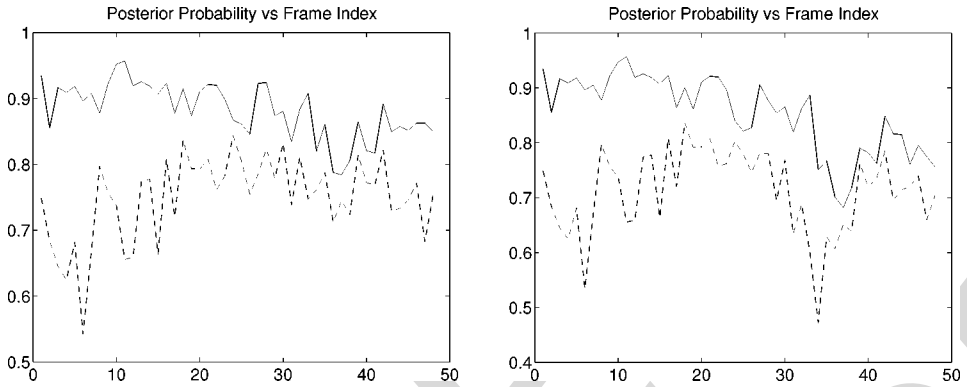

Fig. 5. Posterior probability in a region specified by Σ centered at the mean. The solid and dashed lines are for the true and false hypotheses respectively. Left: $\Sigma = [\infty,\ \infty,\ \infty,\ \infty,\ 5,\ 5]^T$ (equivalent to the *marginal* probability of the translation parameters on a region of size 10×10). Right: $\Sigma = [0.1,\ 0.1,\ 0.1,\ 0.1,\ 5,\ 5]^T$.

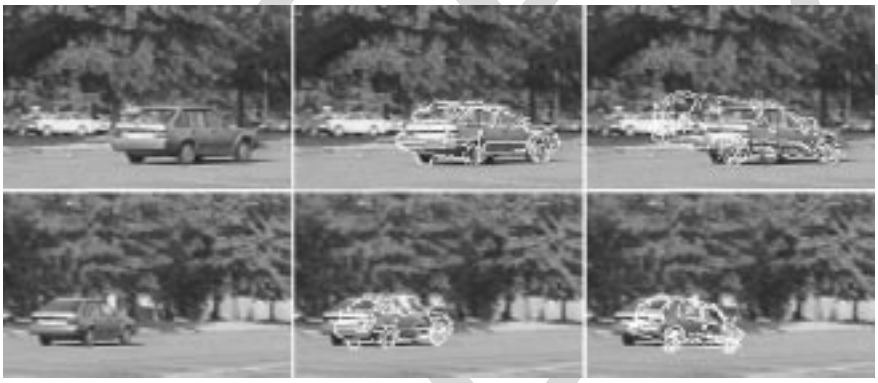

Fig. 6. Left: sample frames of a sequence (Movie 2). Middle and right: tracking results for true and false hypotheses, respectively, (Movies 3, 4).

that the estimates are closer to the ground truth when the hypothesis is true.

Using the verification algorithm, we compute the posterior probability around the mean of the state X_t. Fig. 5 shows the computed P as a function of time t. From the figure, it is obvious that the true hypothesis always gives the higher posterior probability.

B. Vehicle Tracking and Verification

We now present an application to vehicle tracking and verification in real video data, using intensity edges as templates. With real sequences, one needs first to detect the object and rectify the template to the scene. As mentioned in Section III, we assume that this step is done. In our experiments, the algorithm was initialized manually, before invoking the tracking and verification algorithm. The sequences used were acquired by a hand-held video camera. Sample frames of a sequence are shown in Fig. 6 (also Movie 2). In the sequence, the car is receding from the camera, but the camera is subject to zooming and panning, resulting in complex dynamics. This type of sequence may arise from applications involving a moving camera.

We model the motion using the 2-D affine group, and test the two hypotheses against the cluttered sequence. The two candidates are from the previous example (Fig. 1), one true and one false. As an example, Fig. 7 shows samples of the estimated posterior probability densities for the two translation components

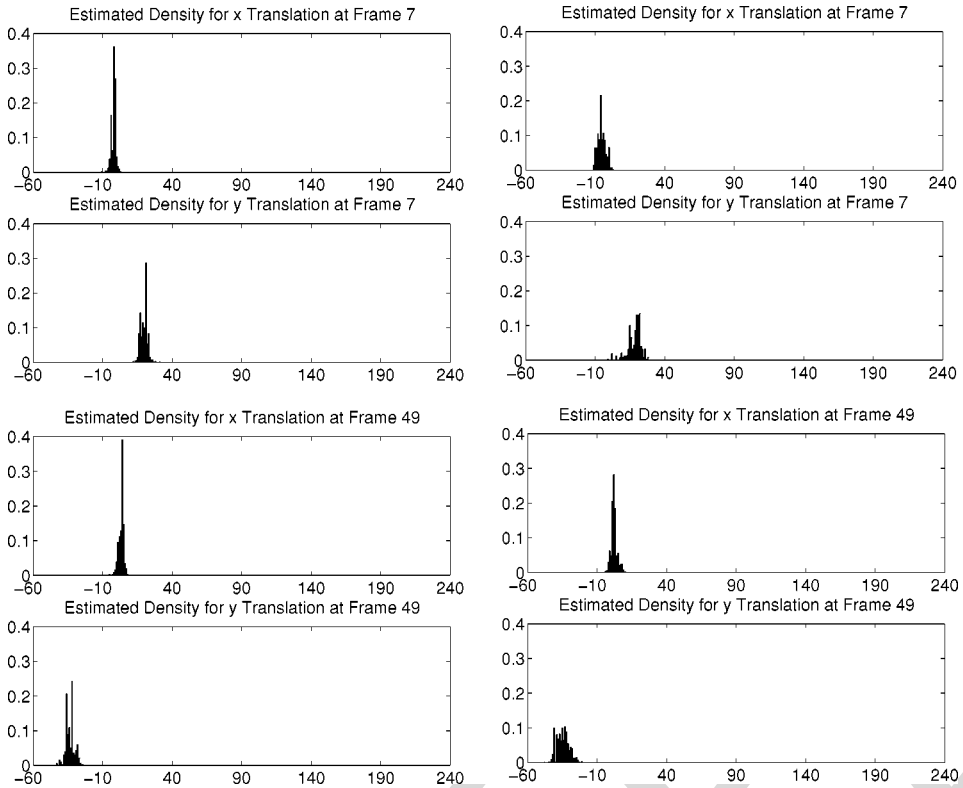

Fig. 7. Estimated densities for translation components at frames 7 and 49. Left: true hypothesis. Right: false hypothesis.

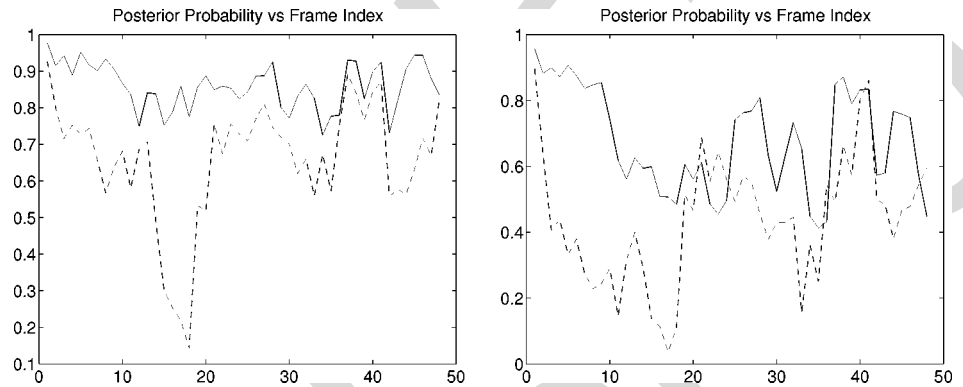

Fig. 8. The posterior probability P for the true (solid) and false (dashed) hypotheses respectively. Left: marginal P in the translation components with $\Sigma = [\infty, \ldots, \infty, 5, 5]^T$. Right: $\Sigma = [0.1, 0.1, 0.1, 0.1, 5, 5]^T$.

of the state vector X_t. The computed posterior probabilities are plotted in Fig. 8. From the figure, it is obvious that the true hypothesis almost always gives the higher posterior probability. If we also use the posterior probability in a retrospective way (consider P up to time t), at any time t there is no difficulty in getting the correct verification result despite the resemblance between the two candidates.

In Fig. 6, we overlap the templates [warped by $E_\pi(X_t)$] onto the original sequence. It is observed that the true hypothesis matches the scene well, while the false one starts to have trouble after first few frames (see also Movies 3, 4). Note that the 2-D affine transform is only an approximation of the vehicle motion in this real video; therefore, even the true template cannot exactly match the scene.

In this experiment, since edges are used, the prediction error is again given by (4).

C. Face/Human Tracking and Verification

Verifying faces in video has broad applications such as surveillance. Unlike the situation for face recognition, surveillance cameras typically have low resolution. Therefore, feature extraction is difficult, and it is desirable to use the image in a holistic way. In the following experiments, we use the intensity face image as the cue for verification. The templates are simply face regions (about 20 $\times$ 18 pixels in size) extracted from the sequence (at a time t not overlapping with the video clip used for tracking and verification). The motion is modeled by the 2-D affine group. A very simple criterion is adopted for computing the prediction error: the *summed absolute difference* (SAD), as used in most block-based compression schemes such as MPEG. Although more complex methods could be used to handle situations such as illumination changes, we found

Fig. 9. Left (Movie 5): sample frames of a sequence. Middle (Movie 6): templates overlaid on the video when the hypotheses are true. Right (Movie 7): when the hypotheses are false.

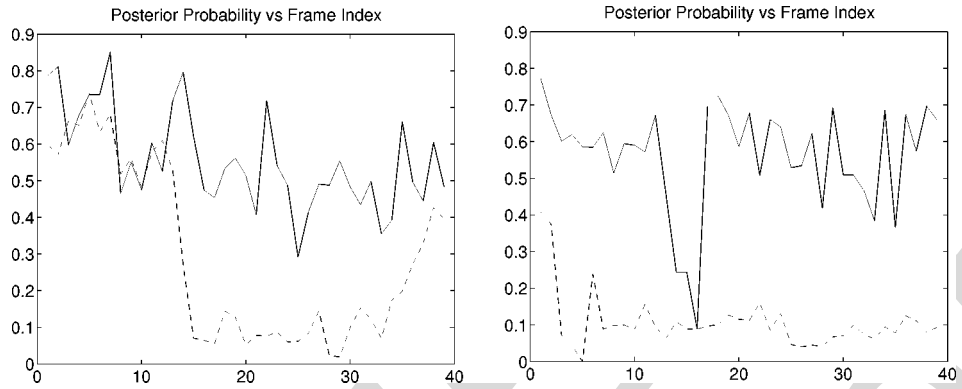

Fig. 10. The posterior probability P for the hypotheses M1 (solid) and M2 (dashed), with $\Sigma = [\infty, \ldots, \infty, 5, 5]^T$. Left: the templates are verified against the right persons. Right: the templates are verified against the wrong persons.

that SAD is easy to compute, and sufficient for the purpose of illustrating how the algorithm works. Of course, the algorithm is capable of using other prediction error criteria.

Fig. 9 (left column) (also Movie 5) shows sample frames of a sequence with two persons moving around, whose face templates are to be verified from the video. In the middle and right columns (also Movie 6, 7), we overlap the templates on the video. For easy visualization, a black block is used for the template corresponding to the face of the man in the white shirt (denoted by M1), and a white block for the template corresponding to the face of the second man (denoted by M2). The middle column illustrates the situation where the algorithm is correctly initialized, meaning that the templates are correctly put on their respective persons. The figure and movies show that tracking is maintained all the time for M1, and is able to recover from occlusion for M2. Fig. 10 (left) shows the computed probability for this case. Note that during the time that M2 is occluded, the probability drops sharply, while M1 maintains high probability across all frames.

The right column in Fig. 9 shows an interesting case: we switch the hypotheses—put the templates on the wrong persons. It is observed that M2 eventually gets dropped to the cluttered background, while M1, first sticking to the wrong person, is attracted to the right person after the men meet. Fig. 10 (right) shows the computed probability for this situation. The curve for M2 (low probability) conveys a lack of confidence, while the curve for M1 shows that, after it is attracted to the right person, the tracker is confident of what it is verifying since the probability is high. It is worth pointing out that during the short period before occlusion occurs, M1 also gives a high probability even though the tracker is on the wrong person. The reason is that during that period, M1 is tracking the face against a clean background (the wall); thus the high probability reflects the notion that the tracker would rather stick to the wrong face than move to the background since the wrong face is at least more face-like than the background. Note that this does not affect verifying which person is M1, since the probability is still lower than when M1 is on the right person [given in Fig. 10 (left)].

This sequence can be thought of as a benign situation except for the occlusion part since the motion is mainly translation and the video quality is good. We now experiment with a challenging sequence of fair visual quality where the motion has large scaling components. Sample frames of the sequence are shown in Fig. 11 (also Movie 8). The task is to verify the man carrying a suitcase as one of two templates (extracted from the two men walking in the middle at a later time in the sequence). Fig. 12 shows an example of the estimated densities of Δa_{11} and Δa_{12}. The computed probability is plotted in Fig. 13. Note that although there are several difficult frames, the algorithm is able to pick the true hypothesis based on the cumulative probability. In the middle and right columns in Fig. 11 (also Movies 9, 10), the template (shown as a white block for easy visualization)

Fig. 11. Left (Movie 8): sample frames of a sequence. Middle and Right: tracking results for true and false hypotheses, respectively (Movies 9, 10).

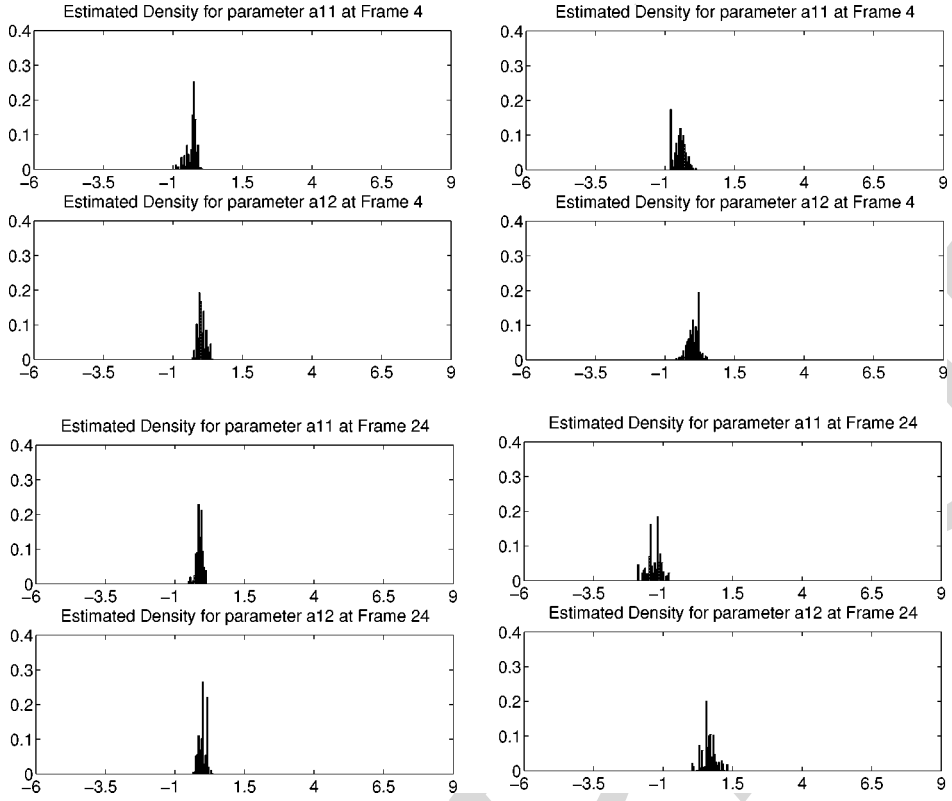

Fig. 12. Posterior densities for Δa_{11}, and Δa_{12} at frames 4 and 24. Left: true hypothesis. Right: false hypothesis. Note that, with the false hypothesis, the densities have not only bad shapes, but also wrong mean values, resulting in the wrong shape and location of the template in Fig. 11 (right).

is overlaid on the sequence. It is obvious that when the hypothesis is true, the algorithm approximately estimates the scaling of the template, while under the false hypothesis, even though the block is deformed so greatly (indicating the competition in trying to verify a person against the wrong template), the template eventually gets stuck in the cluttered background.

Real Application Example: Finally, we use an example to illustrate how the algorithm can be used in surveillance applications. In this example, we assume that we have two cameras at the back and front doors of a building. The camera at the back door captures a person entering the building; sample frames of this sequence are shown in Fig. 14. Later, the camera at the front door finds that two persons are leaving the building, and we want to know which is the person who entered the building a few moments ago. We assume that the cue for tracking and verification is an intensity template, detected by the first camera (for example using background subtraction), as shown in Fig. 14. Notice that in this example, part of the torso is included in the template, since in the video from the front-door camera the faces are too small to support verification (as may be the case in many surveillance applications).

Fig. 15 shows the results of tracking, where we have overlaid the template onto the frames. The left column is the original sequence (Movie 11), and the middle one (Movie 12) is the result when we put the template onto the right person. The right column (Movie 13) is the result when we put the template onto

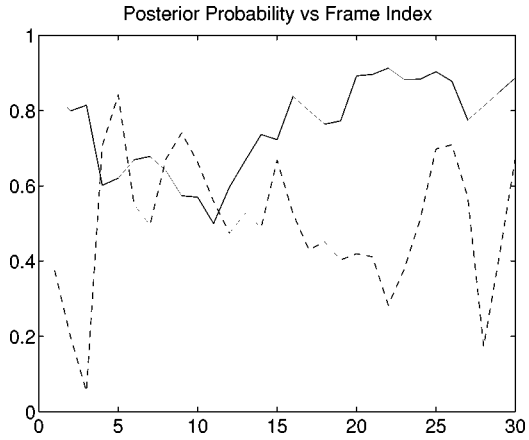

Fig. 13. Posterior probability P for the true (solid) and false (dashed) hypotheses respectively, with $\Sigma = [0.3,\ 0.3,\ 0.3,\ 0.3,\ 4,\ 4]^T$.

the wrong person. Notice that the samples are drawn from a population. When this population covers large X and Y translations (in this example, the population is a zero-mean Gaussian with variance four pixels for the translations), it is found that even though we put the template onto the wrong person, it is eventually attracted to the right person. Thus verification is easily achieved. If we use a population with smaller variances in the translation components (in this example, two pixels), we obtain the results given in Fig. 16, where the tracker sticks to the wrong person. However, we find that the computed posterior probability gives correct verification results as before. The quality of the tracking is best viewed using the accompanying movies.

Some comments are helpful in understanding this example well.

- We are solving a "verification" problem; therefore the algorithm only tells us which of the two persons is more likely the person who entered. It does not judge whether either of the two persons "is" the person who entered (which is a recognition problem).
- This example is a challenging one since the template and the video to verify against are obtained under different lighting conditions (the former is in shadow, the latter in bright sunshine). Thus SAD may not be a good error metric. Even though we use this simple error measure here for illustration purposes, it is unquestionable that a better error metric should be used for robustness and accuracy in real applications.
- We only use image intensity as the cue, but color information can be easily incorporated into the algorithm. For example, one can use color templates instead of the grayscale templates.
- Other variations of this example are also of practical interest. For example, when two persons enter the building, and later one person leaves, the system may be tasked to find who has left. This problem can be solved similarly.
- Although we only use simple dynamics in our experiments, in specific applications, we could do better by using *a priori* information about the problem. For example, in the problem illustrated here, if the cameras are fixed, we know that people leaving the building will be approaching the camera; thus the image of each person in a sequence will be getting larger. This type of information can be used to choose an appropriate dynamic model.
- As pointed out in Section IV-C, although in the experiments we only use posterior probabilities for verification, prediction error is another type of information that is helpful for verification. This type of information is important especially when the candidates are very similar. For instance, in this example, since the sizes of the persons are small, and both of them wear white T-shirts and dark pants, more detailed comparison involving prediction error may be required.

D. Facial Feature Tracking for Verification

Facial feature tracking is useful in applications such as facial expression analysis, face-based person authentication, video-based face recognition, etc. When a short video sequence is available, the face recognition problem can be solved better if temporal information is properly exploited. For example, video-based segmentation can be used to help in face detection, as in [18], [34], [36]. Our algorithm, when applied to facial feature tracking, provides not only tracking but also verification results. This additional gain comes from the fact that one can choose different X_0s for different purposes. If X_0 corresponds to a feature set extracted from the first frame of a sequence, then the algorithm is suitable for *pure tracking*. However, if X_0 represents some templates from a candidate list, then the algorithm is naturally good for *tracking-for-verification*: When a template and the sequence belong to the same person, the tracking results should reflect a coherent motion induced by the same underlying shape; whereas, a more random motion pattern would be observed when a template and the sequence belong to different persons. This idea becomes clear from the following experiments.

Fig. 17 illustrates the facial feature tracking experiments (see also Movies 16, 17). The features to be tracked are defined as a set of Gabor attributes (jets) on a grid. The main motivation behind this is the successful application of Gabor filters to face recognition (e.g., [29], [24], [31]). The motion of facial feature points is modeled by a global 2-D affine transformation (accounting for head motion) plus a local deformation (accounting for residual motion due to inaccuracy of the 2-D affine model and other factors such as facial expressions). Motion of both types is estimated simultaneously by the tracker: global motion is tracked by importance sampling, and residual motion is handled by incorporating local deformation into the measurement likelihood in computing the prediction error. Due to space limit, we refer the readers to [26] for a more complete presentation on facial feature tracking using the proposed method.

E. Image Sequence Stabilization

We now describe a simple experiment showing how the proposed algorithm can be applied to image sequence stabilization (also known as sensor motion compensation). Recall that the algorithm tracks an object through estimating the density of a state vector. If we model the camera motion by a certain transformation group, and use the transformation parameter as the state vector, then motion compensation is solved after the tracking is done. In this experiment, we model the camera motion as a 2-D

Fig. 14. Left: Sample frames from the video taken by the back-door camera. Right: Extracted template.

Fig. 15. Tracking and verification results. Left (Movie 11): sample frames of a sequence. Middle (Movie 12): tracking result when the template is put on the right person. Right (Movie 13): tracking result when the template is put on the wrong person. Note that in this situation, the tracker is attracted to the right person even if it is put on the wrong person, because a sampling population with large variance in the translational components is used. For easier visualization, we use white boundaries to highlight the templates.

Fig. 16. Left (Movie 11): sample frames of a sequence. Middle (Movie 14): results when the template is put on the right person. Right (Movie 15): results when the template is put on the wrong person. Note that in this situation, the tracker is forced to stick to the wrong person by using a sampling population with small variance in the translational components. For easier visualization, we use white boundaries to highlight the templates.

affine transformation. Stabilization is then achieved by the following procedure. First, we extract a set of feature points from the first frame (used as reference frame). Next, we use the algorithm to track the set of feature points in the sequence. Stabilization is then done by warping the current frame with respect to the reference frame using the estimated motion parameter (obtained by evaluating the mean value of the state vector). In this experiment, SAD was again used to measure the prediction error. The SAD was computed on a 5×5 block centered at each feature point.

Fig. 18 illustrates the stabilization results. The top row shows two consecutive frames from the original sequence, with the tracked feature points highlighted. The feature points were detected using a public-domain feature detector [3] based on the KLT algorithm [33]. The bottom row shows the frame differences before and after stabilization. For a better visual effect, see Movie 18 for the original sequence and Movie 19 for the sequence after stabilization.

The stabilization example given here is intended only to illustrate how the proposed algorithm can be applied to tasks

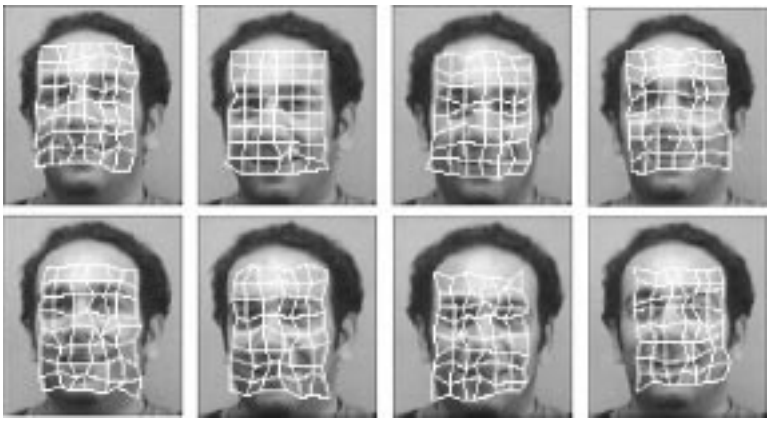

Fig. 17. Tracking results when the template and video belong to the same person (top) and different persons (bottom). See also Movies 16 and 17.

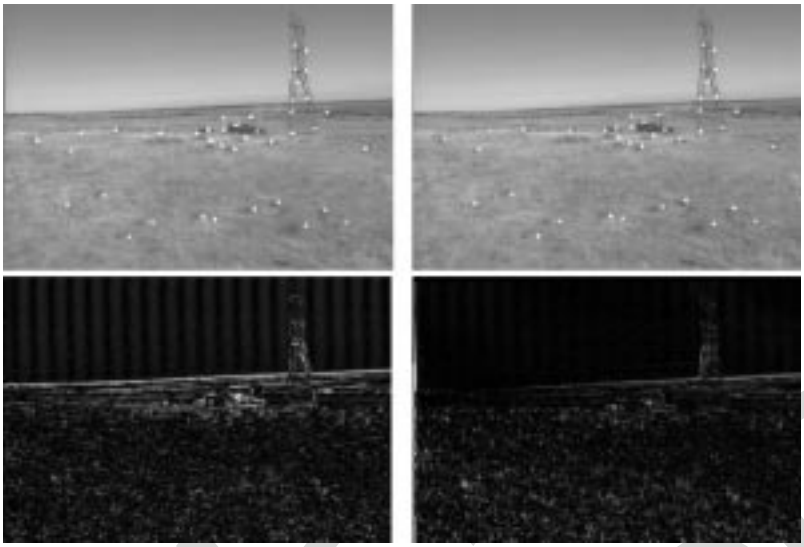

Fig. 18. Image sequence stabilization example. Top: two frames from the original sequence with tracked feature points highlighted. Bottom: frame difference before (left) and after (right) stabilization (gray level re-scaled for easy visualization). See also Movies 18 and 19.

other than tracking, and the resulting stabilization algorithm still has much room for improvement (for example, in the previous experiment, a perspective transformation might be better than 2-D affine, since there is significant 3-D depth variation in the scene), which is not within the scope of this paper. It is also worth pointing out that this solution to stabilization has an important advantage: no feature correspondence is required.

V. Related Work

There has been extensive work in the literature on visual tracking, e.g., [2], [4], [9], [16], [20], [37], [38]. We can only give a few examples here. Reference [20] proposed a contour tracking algorithm using a random sampling method; this was one of the first efforts to use sampling methods in visual tracking, and the approach can be treated as a special realization of the SIS algorithm. In terms of tracking, our approach has extended *beyond* contour tracking, compared with the CONDENSATION algorithm. Of course, a more important aspect is that our approach solves verification in addition to tracking. Also, by using the general formulation in this paper, it is straightforward to incorporate other mature statistical techniques into tracking/verification algorithms. For example, one can easily extend our algorithm by using the generic Monte Carlo algorithm proposed in [27]. Meanwhile, the general formulation makes it obvious to apply the algorithm to other problems such as image sequence stabilization, structure from motion, etc.

Reference [16] proposed a tracking approach based on edge matching. Contours are in a sense features at a lower level compared with intensity images, but at a higher level compared with edges. Thus with contours, efficient computation is easier to achieve. On the other hand, the approach in [16] has the virtue that virtually no model is needed; the algorithm simply works on the sequence (after extracting edges) through matching based on the Hausdorff measure [17]. An eigenspace-based approach was presented in [4] which can directly handle intensity images. To achieve tracking, a training process is needed to construct the eigenspace. In terms of tracking, the proposed approach differs from the previous methods in that it can handle edges, contours, intensity images, or other representations in a uniform way.

There are fewer reports on verification from video. In fact, we are not aware of any other work that explicitly attempts to solve the verification problem *in addition to* tracking. In aforementioned face recognition/verification work, none of the trackers

can give *verification* results as our algorithm does, although we believe that some of them handle the tracking part well. This is the major feature that makes our algorithm different from pure tracking methods.

There is no doubt that verification is an important aspect of video-related applications. When verification is the concern, contours alone are in general insufficient. While edges may be enough for distinguishing, for example, different type of vehicles, they are not expected to work well for verifying more complex objects such as faces. In such cases, we may have to go back to the intensity images (or equivalent transform domain representations). These different representations can be handled uniformly by our algorithm to give tracking *plus* verification results.

Image sequence stabilization is another well-researched problem (e.g., [19], [5], [15], [30], [8], [35]). Generally speaking, image stabilization algorithms can be roughly classified into two categories: feature-based (e.g., [30]) and optical-flow-based (e.g., [8], [35]) approaches. Feature-based approaches depend on the correspondence of feature points between two frames, while optical flow based approaches try to fit the extracted flow to a motion model. Since the flow field is estimated from the whole image, optical flow based approaches tend to be more robust. The use of our algorithm for stabilization is an example of a feature-based approach. But it differs from most feature-based methods such as [30] in that it does not require feature detection except in the first (reference) frame, and thus no feature correspondence is required. The correspondence between features is effectively established by the sampling process: good candidates for correspondence are enforced through importance sampling, and unlikely correspondences are filtered out. Since no feature correspondence is needed, one can increase the number of features to gain robustness, and the only increase in complexity is due to the computation of prediction error (with other feature-based methods, this would cause significant difficulty and complexity in solving for correspondences).

VI. Discussion and Conclusions

This paper presents a generic simultaneous tracking and verification algorithm based on sequential Monte Carlo sampling methods. With this algorithm, applications using edges, contours, intensity images, or other suitable representations can all be unified under the same framework. Experiments on synthetic and real data have been presented. Applications to vehicle/human(face) tracking and verification, facial feature tracking for verification, and image sequence stabilization are illustrated with experiments. The results suggest that the algorithm provides a promising approach to stochastic tracking and verification.

For easier appreciation of the dynamic character of the examples in this paper, a World Wide Web (WWW) site has been established where the sequences referred to in this paper can be watched as MPEG movies.

It is worth pointing out that although in the examples in this paper the motion model is assumed to be 2-D, it need not be stationary in time, meaning that the motion can vary from frame to frame. Thus, this model can describe complex dynamics, such as the one shown in Fig. 6 (Movie 2). Besides, we did not assume any specific system dynamics, other than a simple first-order Gauss–Markov model. In some applications, the system dynamics are known, or can be learned from sample data. In these situations, incorporating the system dynamics should yield better performance than using the simple Gauss–Markov model.

A problem remaining to be solved is dealing with more complex motions. Recall that we deduced the re-parametrization based on the small motion assumption through a Taylor series expansion. Even though (2) is general, it represents a parametric motion model. It may be difficult to obtain a parametric model for complex motions such as those involving 3-D rotation of the object. Fortunately, the parametrization is independent of the algorithm in the sense that the algorithm is applicable as along as a low-dimensional dynamic system is assumed.

The time complexity of the algorithm is another issue that needs to be addressed and that has not been fully investigated yet. However, our experiments suggest that real-time implementation would not be difficult. To give a rough idea about the speed of the algorithm, in the face verification example illustrated in Fig. 11, the algorithm operated at 3 frames/s on an Ultra 5 Sparc workstation for a single hypothesis. The algorithm was able to achieve 4 frames per second on the same workstation for the stabilization example, where image size was 237 $\times$ 348. Since the algorithm has not yet been integrated into a real-time system, the time consumed includes overhead such as reading and writing image files from hard disk, etc. Also, no code optimization of any sort has been performed yet. Therefore, we believe that real-time implementation is possible. Further investigation is needed for accurate characterization of the algorithm's speed performance. Future work also includes efficient real-time implementation of the algorithm.

References

[1] B. Anderson and J. Moore, *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[2] A. Baumberg and D. Hogg, "An efficient method for contour tracking using active shape models," in *Proc. IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, 1994, pp. 194–199.

[3] S. Birchfield. An implementation of the Kanade–Lucas–Tomasi feature tracker. [Online]. Available: http://www.vision.stanford.edu/birch/klt.

[4] M. Black and A. Jepson, "EigenTracking: Robust matching and tracking of articulated objects using a view-based representation," *Int. J. Comput. Vis.*, vol. 20, pp. 63–84, 1998.

[5] P. Burt and P. Anandan, "Image stabilization by registration to a reference mosaic," in *Proc. DARPA Image Understanding Workshop*, 1994, pp. 425–434.

[6] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, pp. 679–698, 1986.

[7] B. P. Carlin, N. Polson, and D. Stoffer, "A Monte Carlo approach to nonnormal and nonlinear state-space modeling," *J. Amer. Statist. Assoc.*, vol. 87, pp. 439–500, 1992.

[8] Z. Duric and A. Rosenfeld, "Stabilization of image sequences," Univ. Maryland, College Park, Tech. Rep. CAR-TR-778, 1995.

[9] D. Freedman and M. Brandstein, "A subset approach to contour tracking in clutter," in *Proc. Int. Conf. Computer Vision*, 1999.

[10] A. E. Gelfand and A. F. M. Smith, "Sampling-based approaches to calculation marginal densities," *J. Amer. Statist. Assoc.*, vol. 85, pp. 398–409, 1990.

[11] C. J. Geyer, "Conditioning in Markov chain Monte Carlo," *J. Comput. Graph. Statist.*, vol. 4, pp. 140–154, 1995.

[12] W. R. Gilks, G. O. Roberts, and E. I. George, "Adaptive direction sampling," *Statistician*, vol. 43, pp. 179–189, 1994.

[13] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *Proc. IEE Radar Signal Processing*, vol. 140, 1993, pp. 107–113.
[14] J. Hammersley and D. Handscomb, *Monte Carlo Methods*. New York: Wiley, 1964.
[15] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P. Burt, "Real-time scene stabilization and mosaic construction," in *Proc. ARPA Image Understanding Workshop*, 1994, pp. 457–465.
[16] D. Huttenlocher, J. Noh, and W. Rucklidge, "Tracking nonrigid objects in complex scenes," in *Proc. Int. Conf. Computer Vision*, 1993.
[17] F. Hausdorff, *Set Theory*, 3rd ed. New York: Chelsea, 1978.
[18] A. J. Howell and H. Buxton, "Toward unconstrained face recognition from image sequences," in *Proc. Int. Conf. Automatic Face and Gesture Recognition*, 1996, pp. 224–229.
[19] M. Irani, B. Rousso, and S. Peleg, "Recovery of ego-motion using image stabilization," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1994, pp. 454–460.
[20] M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density," in *Proc. Eur. Conf. Computer Vision*, 1996.
[21] M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods*. New York: Wiley, 1986.
[22] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *J. Comput. Graph. Statist.*, vol. 5, pp. 1–25, 1996.
[23] A. Kong, J. Liu, and W. Wong, "Sequential imputations and Bayesian missing data problems," *J. Amer. Statist. Assoc.*, vol. 89, pp. 278–288, 1994.
[24] M. Lades, J. Vorbruggen, J. Buhmann, J. Lange, C. v. d. Malsburg, and W. Konen, "Distortion invariant object recognition in the dynamic link architecture," *IEEE Trans. Comput.*, vol. 42, pp. 300–311, 1993.
[25] B. Li and R. Chellappa, "Simultaneous tracking and verification via sequential Monte Carlo methods," in *Proc. IEEE Conf. Computer Vision Pattern Recognit.*, 2000.
[26] ——, "Face verification through tracking facial features," *J. Opt. Soc. Amer. A*, vol. 18, pp. 2969–2981, 2001.
[27] J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *J. Amer. Statist. Assoc.*, vol. 93, pp. 1031–1041, 1998.
[28] C. J. Masreliez, "Approximate non-Gaussian filtering with linear state and observation relations," *IEEE Trans. Automat. Contr.*, vol. 20, pp. 107–110, 1975.
[29] B. S. Manjunath, R. Chellappa, and C. v. d. Malsburg, "A feature based approach to face recognition," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 1992, pp. 373–378.
[30] C. Morimoto and R. Chellappa, "Fast electronic digital image stabilization," in *Proc. Int. Conf. Pattern Recognit.*, 1996, pp. 284–288.
[31] K. Okada, J. Steffens, T. Maurer, H. Hong, E. Elagin, H. Neven, and C. v. d. Malsburg, "The Bochum/USC face recognition system," in *Face Recognition: From Theory to Applications*, H. Wechsler, P. J. Phillips, V. Bruce, F. Soulie, and T. Huang, Eds. Berlin, Germany: Springer-Verlag, 1998.
[32] G. O. Roberts and A. F. M. Smith, "Simple conditions for the convergence of the Gibbs sampler and Hastings–Metropolis algorithm," *Stochast. Proc. Applicat.*, vol. 49, pp. 207–216, 1994.
[33] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1994.
[34] M. Seibert and A. M. Waxman, "Combining evidence from multiple views of 3-D objects," *Proc. SPIE*, vol. 1611, 1991.
[35] S. Srinivasan and R. Chellappa, "Image stabilization and mosaicking using the overlapped basis optical flow field," in *Proc. IEEE Int. Conf. Image Processing*, 1997.
[36] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cogn. Neurosci.*, vol. 3, pp. 72–86, 1991.
[37] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 780–785, 1997.
[38] Y. Yacoob and L. Davis, "Tracking rigid motion using a compact-structure constraint," in *Proc. Int. Conf. Comput. Vis.*, 1999.
[39] S. Zeger and M. R. Karim, "Generalized linear models with random effects: A Gibbs sampling approach," *J. Amer. Statist. Assoc.*, vol. 86, pp. 79–86, 1991.

Baoxin Li (S'97–M'00) received the B.S. and M.S. degrees in electrical engineering from the University of Science and Technology of China in 1992 and 1995, respectively. He received the Ph.D. degree in electrical engineering from the University of Maryland, College Park, in 2000.

He is currently with Sharp Laboratories of America, Inc., Camas, WA, working on multimedia analysis for consumer applications. He was previously with the Center for Automation Research, University of Maryland, working on face and object tracking and verification in video, automatic target recognition, and neural networks. His interests include pattern recognition, computer vision, neural networks, and multimedia processing.

Rama Chellappa (S'78–M'79–SM'83–F'92) received the B.E. (Hons.) degree from the University of Madras, India, in 1975 and the M.E. (Distinction) degree from the Indian Institute of Science, Bangalore, in 1977. He received the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1978 and 1981 respectively.

Since 1991, he has been a Professor of electrical engineering and an affiliate Professor of computer science at the University of Maryland, College Park. He is also affiliated with the Center for Automation Research (Associate Director) and the Institute for Advanced Computer Studies. Prior to joining the University of Maryland, he was an Associate Professor and Director of the Signal and Image Processing Institute with the University of Southern California, Los Angeles. Over the last 19 years, he has published numerous book chapters and peer-reviewed journal and conference papers. Several of his journal papers have been reproduced in Collected Works published by IEEE Press, IEEE Computer Society Press and MIT Press. He has edited a collection of *Papers on Digital Image Processing* (published by IEEE Computer Society Press), co-authored a research monograph on *Artificial Neural Networks for Computer Vision* (with Y. T. Zhou) published by Springer Verlag, and co-edited a book on *Markov Random Fields* (with A. K. Jain) published by Academic Press. He was co-Editor-in-Chief of *Graphical Models and Image Processing*. His current research interests are image compression, automatic target recognition from stationary and moving platforms, surveillance and monitoring, automatic design of vision algorithms, synthetic aperture radar image understanding, and commercial applications of image processing and understanding.

Dr. Chellappa has served as an associate editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IMAGE PROCESSING, NEURAL NETWORKS. He also served as a member of the IEEE Signal Processing Society Board of Governors during 1996–1999. He is currently serving as the Editor-in-Chief of IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He has received several awards, including the 1985 NSF Presidential Young Investigator Award, a 1985 IBM Faculty Development Award, the 1991 Excellence in Teaching Award from the School of Engineering at USC, and the 1992 Best Industry Related Paper Award from the International Association of Pattern Recognition (with Q. Zheng). He has been recently elected as a distinguished Faculty Research Fellow (1996–1998) at the University of Maryland. He is a Fellow of the International Association for Pattern Recognition. He has served as a General and Technical program Chair for several IEEE international and national conferences and workshops.