Real-time Speaker Tracking Using Particle Filter Sensor Fusion

Yunqiang Chen and Yong Rui

***Abstract*— Sensor fusion for object tracking has become an active research direction during the past few years. But how to do it in a robust and principled way is still an open problem. In this paper, we propose a new fusion framework that combines both the bottom-up and top-down approaches to probabilistically fuse multiple sensing modalities. At the lower level, individual vision and audio trackers are designed to generate effective proposals for the fuser. At the higher level, the fuser performs reliable tracking by verifying hypotheses over multiple likelihood models from multiple cues. Different from the traditional fusion algorithms, the proposed framework is a closed-loop system where the fuser and trackers coordinate their tracking information. Furthermore, to handle non-stationary situations, the proposed framework evaluates the performance of the individual trackers and dynamically updates their object states. We present a real-time speaker tracking system based on the proposed framework by fusing object contour, color and sound source location. We report robust tracking results.**

***Index Terms*— Sensor fusion, tracking, particle filter, real-time.**

I. Introduction

Distributed meetings and lectures are gaining significant attentions during the past few years [1], [2]. A key technology component in those systems is a reliable speaker tracking module. For instance, if the system knows the speaker location, it can point a camera at the speaker dynamically so that remote audience can have a zoomed-in view of the active speaker. There are commercial video conferencing systems, e.g., [3], that provide speaker tracking based on audio sound source localization (SSL). While this is much better than using a static camera, it is far from sufficient. SSL is a good speaker detector but not a good speaker tracker especially when the person is not constantly talking. Reliable speaker tracking therefore involves high-performance audio-based SSL, vision-based person tracking and sensor fusion techniques. We reported our audio- and vision-based tracking techniques in [4] and [5] respectively. In this paper, we focus on novel sensor fusion frameworks.

In general, there are two existing paradigms for sensor fusion: bottom-up and top-down. Both paradigms have a fuser and multiple sensors. Throughout the paper, we use the term "sensor" in a generalized way. It represents a *logical sensor* instead of a *physical sensor*. For example, both contour sensor and color sensor are based on the same physical sensor – a video camera. Depending on the complexity of the sensor algorithms, the sensors can perform different tasks. For example, some perform tracking and are called *trackers*. Others perform verification (i.e., computing the likelihood of a giving hypothesis) and are called *verifiers*.

The bottom-up paradigm starts from the sensors. Each sensor has a tracker and it tries to solve the *inverse* problem – estimating the *unknown* object state (e.g. object location and orientation) based on the sensory data. Once the individual

Manuscript received March, 2003; revised November 11, 2003.

Yunqiang Chen is with Siemens Corporate Research.

Yong Rui is with Microsoft Research.

tracking results are available, distributed sensor networks [6] or graphical models [7] are used to fuse them together to generate a more accurate and robust result in the fuser. To make the inverse problem tractable, assumptions are typically made in the trackers and the fuser, e.g., system linearity and Gaussianality are assumed in the Kalman tracker [8] and the fuser [6]. While these assumptions make the problem tractable, they inherently hinder the robustness of the bottom-up scheme. For example, the Gaussian assumption of noise is almost never true in real life, but the bottom-up scheme does not have a mechanism to detect and correct the possible errors caused by the simplified assumptions.

The top-down paradigm, on the other hand, emphasizes on the *top* – it has an intelligent fuser but simple sensors (i.e., verifiers) [9], [10]. It tries to achieve tracking by solving the *forward* problem – evaluating the likelihood of a set of *given* hypotheses using the sensory data. First, the fuser generates a set of hypotheses (also called particles – we use them interchangeably in the paper) to cover the possible state space. All the hypotheses are then sent down to the sensor for verification. The sensors compute the likelihood of each hypothesis and report back to the fuser. The fuser then uses the weighted hypotheses to estimate the distribution of the object state. Note that it is usually much easier to verify a given hypothesis than to solve the inverse tracking problem (as in the bottom-up paradigm). Therefore, more complex models (e.g., non-linear and complex object shape models) can be used in the top-down paradigm. This in turn results in more robust tracking. There is, however, inefficiency with this paradigm. Because the sensors have verifiers instead of trackers, they do not help the fuser to generate good hypotheses. The hypotheses are semi-blindly generated from the transition prior (i.e., motion prediction) in [9]. When a meeting participant can move freely as in our scenario, a large number of hypotheses are needed in order to cover all the possible state space and that costs expensive computation. Many hypotheses can land on low-likelihood regions thus wasted [11]. In [12], an external tracking module is used to guide the hypothesis generation. But the whole system then depends on the robustness of the external module.

To summarize, the bottom-up paradigm provides fast best-effort tracking, but is at the expense of simplified assumptions. On the other hand, the top-down paradigm does not require simplified assumptions but needs expensive computation because the hypotheses can be very inefficient. In this paper, we propose a new fusion framework that integrates the two paradigms to achieve robust real-time tracking. First, it is a closed-loop architecture where the fuser and sensors interact to exchange tracking information. For example, the fuser uses trackers' outputs to construct effective hypotheses, and trackers use fuser's output to guide their own tracking. Second, to effectively utilize multiple sensors, an evaluation and adaptation scheme is proposed to evaluate the reliability of various trackers. Different trackers contribute differently based on their reliability. For failing trackers, they are re-initialized. This adaptation scheme greatly enhances the system's ability to handle non-stationary situations.

The rest of the paper is organized as follows. We first present a proposal-centric view of particle filtering in Section II, setting the stage for the new sensor fusion framework. In Section III, we present the complete fusion algorithm and summarize its characteristics. To apply the new algorithm to real-time speaker tracking, three trackers based on head contour, color, and sound source localization are designed in Section IV. Together, they generate good proposals for the fuser. In Section V, we discuss the verifiers and their likelihood models based on which the hypotheses are evaluated. In Section VI, we further discuss the adaptation scheme: how to evaluate and adapt trackers to handle non-stationary environment. In Section VII, we conduct experiments and report

robust tracking results on challenging real-world sequences. We give concluding remarks in Section VIII.

II. Generic Particle Filtering

In this section we discuss the general particle filtering technique, setting the stage for the sensor fusion framework proposed in the next section. In CONDENSATION [13], extended factored sampling is used to explain how the particle filter works. Even though easy to follow, it obscures the role of proposal distributions. In this section, we present an alternative formulation of the particle filtering theory that is centered around proposal distributions. Proposal distributions can be used both to improve particle filter's performance and to provide a principled way for sensor fusion.

Let $X_{0:t}$ represent the object states (e.g. object position and size) that we are interested in and $Z_{0:t}$ represent the observation (e.g. audio signal or video frame) from time 0 to t.

A non-parametric way to represent a distribution is to use particles drawn from the distribution. For example, we can use the following point-mass approximation to represent the posterior distribution of X:

$$\hat{p}(X_{0:t}|Z_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta_{X_{0:t}^{(i)}}(dX_{0:t}) \tag{1}$$

where δ is the Dirac delta function, and particles $X_{0:t}^{(i)}$ are drawn from $p(X_{0:t}|Z_{1:t})$. The approximation converges in distribution when N is sufficiently large [14], [11]. This particle-based distribution estimation is, however, only of theoretical significance. In reality, the posterior distribution is the one that needs to be estimated, thus not known. Fortunately, we can instead sample the particles from a known proposal distribution $q(X_{0:t}|Z_{1:t})$ and still be able to compute $p(X_{0:t}|Z_{1:t})$.

Definition 1 [15]: A set of random samples $\{X_{0:t}^{(i)}, w_{0:t}^{(i)}\}$ drawn from a distribution $q(X_{0:t}|Z_{1:t})$ is said to be properly weighted with respect to $p(X_{0:t}|Z_{1:t})$ if for any integrable function $g()$ the following is true:

$$E_p(g(X_{0:t})) = \lim_{N\to\infty}\sum_{i=1}^{N} g(X_{0:t}^{(i)})w_{0:t}^{(i)} \tag{2}$$

Furthermore, as N tends to infinity, the posterior distribution p can be approximated by the properly weighted particles drawn from q [15], [11]:

$$\hat{p}(X_{0:t}|Z_{1:t}) = \sum_{i=1}^{N} w_{0:t}^{(i)} \cdot \delta_{X_{0:t}^{(i)}}(dX_{0:t}) \tag{3}$$

$$\tilde{w}_{0:t}^{(i)} = \frac{p(Z_{1:t}|X_{0:t}^{(i)})p(X_{0:t}^{(i)})}{q(X_{0:t}^{(i)}|Z_{1:t})} \tag{4}$$

$$w_{0:t}^{(i)} = \frac{\tilde{w}_{0:t}^{(i)}}{\sum_{i=1}^{N}\tilde{w}_{0:t}^{(i)}} \tag{5}$$

where $\tilde{w}_{0:t}^{(i)}$ and $w_{0:t}^{(i)}$ are the un-normalized and normalized particle weights.

In order to propagate the particles $\{X_{0:t}^{(i)}, w_{0:t}^{(i)}\}$ through time, it is beneficial to develop a recursive calculation of the weights. This can be obtained straightforwardly by considering the following two facts:

1) Current states do not depend on future observations. That is,

$$q(X_{0:t}|Z_{1:t}) = q(X_{0:t-1}|Z_{1:t-1})q(X_t|X_{0:t-1}, Z_{1:t})$$

2) The system state is a Markov process and the observations are conditionally independent given the states [13], [11], i.e.:

$$\begin{aligned} p(X_{0:t}) &= p(X_0)\prod_{j=1}^{t} p(X_j|X_{j-1}) \\ p(Z_{1:t}|X_{0:t}) &= \prod_{j=1}^{t} p(Z_j|X_j) \end{aligned} \tag{6}$$

Substituting the above two equations into Equation (4), we

obtain the recursive estimate for the weights:

$$\begin{aligned}\tilde{w}_t^{(i)} &= \frac{p(Z_{1:t}|X_{0:t}^{(i)})p(X_{0:t}^{(i)})}{q(X_{0:t-1}^{(i)}|Z_{1:t-1})q(X_t^{(i)}|X_{0:t-1}^{(i)}, Z_{1:t})} \\ &= \tilde{w}_{t-1}^{(i)} \frac{p(Z_{1:t}|X_{0:t}^{(i)})p(X_{0:t}^{(i)})}{p(X_{0:t-1}^{(i)})p(Z_{1:t-1}|X_{0:t-1}^{(i)})q(X_t^{(i)}|X_{0:t-1}^{(i)}, Z_{1:t})} \\ &= \tilde{w}_{t-1}^{(i)} \frac{p(Z_t|X_t^{(i)})p(X_t^{(i)}|p(X_{t-1}^{(i)})}{q(X_t^{(i)}|X_{0:t-1}^{(i)}, Z_{1:t})} \end{aligned} \quad (7)$$

Note that the particles are now drawn from the proposal distribution $q(X_t|X_{0:t-1}, Z_{1:t})$ instead of from the posterior p. To summarize, the particle filtering process has three steps:

1. Sampling step: N particles $X_t^{(i)}, i = 1, ..., N$ are sampled from the proposal function $q(X_t|X_{0:t-1}, Z_{1:t})$.

2. Measurement step: Compute the particle weights using Equation (7).

3. Output step: The weighted particles can be readily used as the tracking results. The conditional mean of X_t can be computed using Equation (2) with $g_t(X_t) = X_t$, and conditional covariance of X_t can be computed using Equation (2) with $g_t(X_t) = X_t X_t^T$.

This proposal-centric view sheds new lights on the role of the proposal distribution in the particle filtering process. It provides a way to guide the particle generation. In practice, there are infinite number of choices for the proposal distribution, as long as its support includes that of the posterior distribution. But the quality of proposals can differ significantly. For example, poor proposals (far different from the true posterior) will generate particles that have negligible weights, thus wasted. On the other hand, particles generated from good proposals (similar to the true posterior) are highly effective. Choosing the right proposal distribution is therefore of great importance. Indeed, the proposal is not only at the center of the particle filtering process, but also provides a principled way to perform sensor fusion as explained in next section.

III. Sensor Fusion

Good proposals generate effective particles. This is especially important when we process multiple sensors and the problem state space has high dimensionality. In the context of tracking, various approaches have been proposed to obtain more effective proposals than the transition prior (i.e., $p(X_t|X_{t-1})$) [13]. If there is only a single sensor, an auxiliary Kalman-filter tracker can be used to generate the proposal [16]. When multiple sensors are available, a master-slave approach is proposed in [12], where a slave tracker (color-blob tracker) is used to generate proposals for the master trackers (a particle-based contour tracker). While this approach achieves better results than the single sensor approaches, its master-slave structure breaks the symmetry between trackers. Furthermore, because the slave tracker is not included in the overall observation likelihood model, it may discard complementary and important tracking information from the slave tracker [12].

In this paper, we present a two-level closed-loop particle filter architecture for sensor fusion, with the proposal distribution being the focal point. It integrates the benefits of both the bottom-up and top-down paradigms. For robustness, multiple complementary sensory data are utilized at both levels. At the lower level, individual trackers based on different cues perform independent tracking and report tracking results up to the fuser (Section IV). At the upper level, the fuser constructs an informed proposal by integrating tracking results from all the trackers. Particles are sampled from this proposal and sent down to the verifiers to compute their likelihood (Section V). The set of evaluated particles constitute a good estimate of the posterior [11]. The complete algorithm for the proposed sensor fusion framework is given as follows (refer to Figure 1):

1. Tracking by individual trackers: Using appropriate assumptions (e.g., Gaussianality and linearity), each tracker generates fast but perhaps less robust tracking results

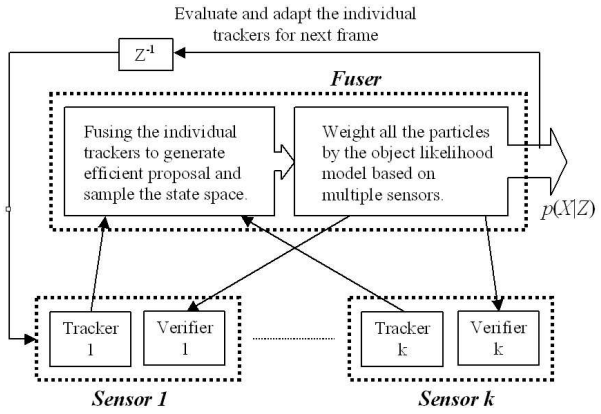

Fig. 1. Fusion diagram.

$q_k(X_t^k|X_{0:t-1}^k, Z_{1:t}^k)$, where k is the index for individual trackers. Different trackers are designed in Section IV.

2. Generating proposal distribution: The fuser integrates the tracking results from multiple trackers to form a mixture of Gaussian distribution as the final proposal:

$$q(X_t|X_{0:t-1}, Z_{1:t}) = \sum_k \lambda_k \cdot q_k(X_t^k|X_{0:t-1}^k, Z_{1:t}^k)$$

where λ_k is the reliability of tracker k and is estimated dynamically in Section VI.

Note that because the final proposal is a mixture of all the individual proposals, our proposed algorithm is robust even when some of the trackers fail. In fact, as long as one of the individual proposals covers the true object state, particles will be generated in the neighborhood of the true state and will get high likelihood score in step 3, thus keeping track of the object.

3. Generating particles and weights: Particles are sampled from the proposal distribution $q(X_t|X_{0:t-1}, Z_{1:t})$ and then sent down to the verifiers to compute their weights:

$$\tilde{w}_t^{(i)} = \tilde{w}_{t-1}^{(i)} \frac{p(Z_t|X_t^{(i)})p(X_t^{(i)}|p(X_{t-1}^{(i)})}{q(X_t^{(i)}|X_{0:t-1}^{(i)}, Z_{1:t})} \quad (8)$$

Assuming independence between the likelihoods from different verifiers, the overall likelihood is:

$$p(Z_t|X_t^{(i)}) = \prod_k p(Z_t^k|X_t^{(i)}) \quad (9)$$

The set of weighted particles can be readily used as the estimate of the posterior distribution (Step 3 in Section II).

Note that each sensor has both a tracker and a verifier. The tracker tries to solve the inverse problem efficiently. Small errors are therefore allowed and can be corrected later by the fuser and verifier. Simplifications (e.g., constant object color histogram or Gaussianality) are usually assumed in the tracker to ensure efficiency. The verifier, on the other hand, only needs to verify a given hypothesis, which is much easier than solving the inverse problem. More comprehensive and accurate likelihood models $p(Z_t|X_t)$ can therefore be exploited in the verifier (see Section V). The separation of tracker and verifier strikes a good balance between efficiency and robustness.

4. Adapting the trackers: To handle non-stationary situations and potential mis-track, individual trackers need feedback from the fuser. Object states (e.g., position and size) and attributes (e.g., color histogram) are updated dynamically based on the fuser's estimation of the posterior. The reliability of each tracker is also evaluated based on the performance of the corresponding proposal. More reliable trackers will contribute more to the proposal functions and unreliable trackers will be reinitialized. We discuss this in detail in Section VI.

The above two-level closed-loop sensor fusion framework is a general framework for combining different cues, individual trackers and high level object likelihood modeling together. It is more robust than the bottom-up paradigm because it uses multiple hypotheses and verifies based on more accurate object model. It is computationally more effective than the top-down paradigm because it starts with good proposal distributions. It is also more reliable than both paradigm because it is a closed-loop system where object states and attributes are dynamically updated. In the following sections, we apply this fusion framework to real-time speaker tracking. We describe three basic trackers in Section IV. We discuss the verifiers in Section V. We describe how to update individual trackers in

Section VI.

IV. Individual Trackers

Although the verification process can correct some tracking errors,it is desirable and effective if the trackers can provide accurate results in the first place. In this section, we design three trackers based on complementary cues. According to the set theory, every closed set (e.g., an object) can be decomposed into two disjoint sets: the boundary and the interior [17]. Since these two sets are complementary, we develop two vision-based trackers that use two complementary cues (object contour and object interior color) to track human heads. We also develop an audio-based SSL tracker which further complements the vision-based trackers.

We approximate a human head as a vertical ellipse with a fixed aspect ratio of 1.2: $X_t = [x_t^c, y_t^c, \alpha_t]$, where (x_t^c, y_t^c) is the center of the ellipse, and α_t is the major axis of the ellipse. In Figures 4 and 5, we represent it with a bounding box. A tracker estimates its belief of the object state X_t^k based on its own observation Z_t^k. Note that it is not required for all the trackers to estimate all the elements in the state vector X_t. For example, while the contour tracker estimates all $[x_t^c, y_t^c, \alpha_t]$, the SSL tracker only estimates x_t^c.

A. The contour tracker

For each frame we use a hidden Markov model (HMM) to find the best contour $\mathbf{s}^*$. We then use the unscented Kalman filter (UKF) [5] to track object state X_t over time. Note the notations used in this section. In HMM, people use $\mathbf{s}^*$ to represent best states and in UKF people use Y_t to represent measurements. We follow these conventions, but we would like to point out that in our context $\mathbf{s}^*$ and Y_t are the same entity which represents the best detected contour. We summarize the contour tracker algorithm as follows [5]:

1. Measurement collection: At time t, edge detection is conducted along the normal lines of the predicted contour location (see Figure 2). We use $Z_t^1 = \{\mathbf{z}_\phi, \phi \in [1, M]\}$ to denote the edge intensity observation, where the superscript "1" in Z_t^1 represents this is the first sensor (i.e., contour sensor). The solid curve is the predicted contour. The dashed curve is the true contour that we want to find. The term $s_\phi, \phi \in [1, M]$ represents the true contour point on the ϕth normal line. Each normal line has $[-N, N] = 2N + 1$ pixels. Because of background clutter, there can be multiple edges on each normal line.

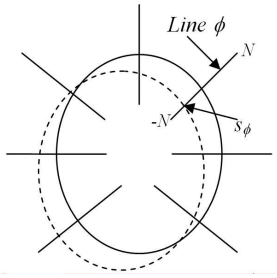

Fig. 2. Illustration of parametric contour tracking: *The solid curve is the predicted contour. The dashed curve is the true contour that we want to find. $s_\phi, \phi \in [1, M]$ is the true contour point on the ϕth normal line, which can be found efficiently by a hidden Markov model.*

2. Contour detection using an HMM: HMM usually is used in the temporal domain (e.g., speech recognition). Here we use it to model the contour smoothness constraint in the spatial domain. Specifically, the hidden states are the true contour position s_ϕ on each normal line. Our goal is to estimate the hidden states based on our observations. An HMM is specified by the likelihood model $p(\mathbf{z}_\phi | s_\phi)$ and the transition probabilities $p(s_\phi | s_{\phi-1})$ which we discuss next.

3. Likelihood model: With the assumption that background clutter is a Poisson process with density γ and the detection error is normally distributed as $N(0, \sigma_z)$, the likelihood that s_ϕ is a contour is given by [13]:

$$p(\mathbf{z}_\phi | s_\phi) \propto 1 + \frac{1}{\sqrt{2\pi}\sigma_z q \gamma} \sum_{m=1}^{J_\phi} \exp(-\frac{(z_m - s_\phi)^2}{2{\sigma_z}^2}) \quad (10)$$

where J_ϕ is the number of detected edges on line ϕ and q is the prior probability of the contour not being detected.

4. Transition probabilities: The state transition probabilities

encodes the spatial dependencies of the contour points on neighboring normal lines. In Figure 2, we can see that the true contour points on adjacent normal lines tend to have similar displacement from the predicted position (i.e., the center of each normal line). This is the contour smoothness constraint and can be captured by transition probabilities $p(s_\phi|s_{\phi-1})$, which penalizes sudden changes between neighboring contour points:

$$p(s_\phi|s_{\phi-1}) = c \cdot e^{-(s_\phi - s_{\phi-1})^2/\sigma_s^2} \quad (11)$$

where c is a normalization constant and σ_s is a predefined constant that regulates the contour smoothness.

5. Optimization: Given the observation sequence $Z_t^1 = \{\mathbf{z}_\phi, \phi \in [1, M]\}$ and the transition probabilities $a_{i,j} = p(s_{\phi+1} = j|s_\phi = i), i, j \in [-N, N]$, the best contour can be obtained by finding the most likely state sequence $\mathbf{s}^*$ using the Viterbi algorithm [18]:

$$\mathbf{s}^* = \arg\max_{\mathbf{s}} P(\mathbf{s}|Z_t^1) = \arg\max_{\mathbf{s}} P(\mathbf{s}, Z_t^1)$$

6. Tracking over time using UKF: UKF extends the regular Kalman filter [8] to handle non-linear systems. Let the system be $X_t = f(X_{t-1}, d_t)$ and $Y_t = g(X_t, v_t)$, where d_t and v_t are zero-mean Gaussian noises. Let $X_{t|t-1} = E(f(X_{t-1}, d_t))$ and $Y_{t|t-1} = E(g(f(X_{t-1}, d_t), v_t))$. The system state X_t can be estimated as:

$$\hat{X}_t^1 = X_{t|t-1} + K_t \cdot (Y_t - Y_{t|t-1})$$

where $Y_t = \mathbf{s}^* = [s_1, ..., s_\phi, ..., s_M]^T$ is the measurement and K_t is the Kalman gain [5]. We use the Langevin process [19] to model the object dynamics:

$$X_t = f(X_{t-1}, d_t) = \begin{bmatrix} 1 & \tau \\ 0 & a \end{bmatrix} \begin{bmatrix} X_{t-1} \\ \dot{X}_{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix} d_t \quad (12)$$

where $a = \exp(-\beta_\theta \tau)$, $b = \bar{v}\sqrt{1-a^2}$. β_θ is the rate constant, d_t is a thermal excitation process drawn from Gaussian distribution $N(0, Q)$, τ is the discretization time step and $\bar{v}$ is the steady-state root-mean-square velocity.

The observation function $g() = [g_1(), ..., g_\phi, ..., g_M()]^T$ represents the relationship between the observation $Y_t = \mathbf{s}^* = [s_1, ..., s_\phi, ..., s_M]^T$ and the state X_t. Let $[x_\phi, y_\phi]$ be line ϕ's center point, and let the intersection of the ellipse X_t and the normal line ϕ be point P_ϕ, the physical meaning of s_ϕ is the distance between P_ϕ and $[x_\phi, y_\phi]$. Further let angle θ_ϕ be line ϕ's orientation, and let $x' = x_\phi - x_t^c$, $y' = y_\phi - y_t^c$, $\alpha = \alpha_t$, and $\beta = \alpha_t/1.2$. We can derive the following relationship between s_ϕ and X_t:

$$s_\phi = g_\phi(X_t, v_t) = v_t + \frac{-\left[\frac{x'\cos\theta_\phi}{\alpha^2} + \frac{y'\sin\theta_\phi}{\beta^2}\right]}{\left[\frac{\cos^2\theta_\phi}{\alpha^2} + \frac{\sin^2\theta_\phi}{\beta^2}\right]} + \frac{\sqrt{\left[\frac{x'\cos\theta_\phi}{\alpha^2} + \frac{y'\sin\theta_\phi}{\beta^2}\right]^2 - \left[\frac{\cos^2\theta_\phi}{\alpha^2} + \frac{\sin^2\theta_\phi}{\beta^2}\right]\left[\frac{x'^2}{\alpha^2} + \frac{y'^2}{\beta^2} - 1\right]}}{\left[\frac{\cos^2\theta_\phi}{\alpha^2} + \frac{\sin^2\theta_\phi}{\beta^2}\right]} \quad (13)$$

Because the $g()$ is nonlinear, unscented transformation is used to estimate the K_t, $X_{t|t-1}$ and $Y_{t|t-1}$ (see [5]).

A Gaussian distributed proposal function can be formed based on the contour tracker:

$$q_1(X_t^1|X_{t-1}^1, Z_t^1) = N(\hat{X}_t^1, \hat{\Sigma}_t^1) \quad (14)$$

where $\hat{\Sigma}_t^1$ is the Kalman filter's covariance matrix [8].

B. The color tracker

Object interior region properties complement its contour in tracking. Color based tracking has been widely used in the literature. We adopt the Meanshift algorithm [20] in our system as the color tracker. It assumes that the color histogram of the target object h_{obj} is stable and a recursive gradient decent searching scheme is used to find the region that is most similar to h_{obj}.

To track the object in the current frame, previous frame's state is used as an initial guess, i.e., $\hat{X}_0 = X_{t-1}^2$, where the

superscript "2"in X_{t-1}^2 means this is the second tracker in the paper, i.e., $k = 2$. The following steps are used to find the new object state X_t^2:

1) Let l index the Meanshift iterations. Set $l = 0$.
2) Compute the color histogram at $\hat{X}_l$: $h_{\hat{X}_l}$. Compute the similarity between $h_{\hat{X}_l}$ and the target using the Bhattacharyya coefficient $\rho[h_{obj}, h_{\hat{X}_l}]$ [20].
3) Compute the color histogram gradient and move the searching window to the new location $\hat{X}_N$ using the Meanshift analysis [20].
4) Compute the color histogram at $\hat{X}_N$: $h_{\hat{X}_N}$. Compute the similarity between $h_{\hat{X}_N}$ and the target using the Bhattacharyya coefficient $\rho[h_{obj}, h_{\hat{X}_N}]$
5) If $\rho[h_{obj}, h_{\hat{X}_N}] > \rho[h_{obj}, h_{\hat{X}_l}]$, goto step 6.
   Else, let $\hat{X}_N = (\hat{X}_l + \hat{X}_N)/2$ and goto step 4.
6) If $\|\hat{X}_N - \hat{X}_l\| < \epsilon$, stop. Otherwise, let $l = l + 1$ and $\hat{X}_l = \hat{X}_N$ and go to Step 2.

When the algorithm converges, $\hat{X}_l$ is the new estimate of the object state $\hat{X}_t^2$. We can then form the second proposal function based on the color tracker:

$$q_2(X_t^2 | X_{t-1}^2, Z_t^2) = N(\hat{X}_t^2, \hat{\Sigma}_t^2) \tag{15}$$

where $\hat{\Sigma}_t^2$ represents the uncertainty of the Meanshift color tracker.

C. The SSL tracker

Vision-based tracker can only provide locations of people. It is the audio-based tracker that identifies which particular person is speaking. But the audio-based trackers have their own limitations: it is quite difficult for them to estimate all the 3 elements in the object state. Fortunately, in our particular application of meetings and lectures, the system cares the most about x_t^c, the horizontal location of the speaker. This simplifies our SSL tracker design – we only need to have two microphones to estimate x_t^c. Let $s(t)$ be the speaker's source signal, and $x_1(t)$ and $x_2(t)$ be the signals received by the two microphones, we have:

$$\begin{aligned} x_1(t) &= s(t - D) + h_1(t) * s(t) + n_1(t) \\ x_2(t) &= s(t) + h_2(t) * s(t) + n_2(t) \end{aligned} \tag{16}$$

where D is the time delay between the two microphones, $h_1(t)$ and $h_2(t)$ represent reverberation, and $n_1(t)$ and $n_2(t)$ are the additive noise. Assuming the signal and noise are uncorrelated, D can be estimated by finding the maximum cross correlation between $x_1(t)$ and $x_2(t)$:

$$\begin{aligned} D &= \arg\max \hat{R}_{x_1 x_2}(\tau) \\ \hat{R}_{x_1 x_2}(\tau) &= \frac{1}{2\pi} \int_{\pi}^{-\pi} W(\omega) X_1(\omega) X_2^*(\omega) e^{j\omega\tau} d\omega \end{aligned} \tag{17}$$

where $X_1(\omega)$ and $X_2(\omega)$ are the Fourier transforms of $x_1(t)$ and $x_2(t)$, $\hat{R}_{x_1 x_2}(\tau)$ is the cross correlation of $x_1(t)$ and $x_2(t)$, and $W(w)$ is a frequency weighting function [4].

Once the time delay D is estimated from the above procedure, the horizontal sound source direction x_t^c can be easily estimated given the microphone array's geometry. Let the two microphones be at positions A and B, and the middle point between them be position O. Let the source be at location S, as illustrated in Figure 3.

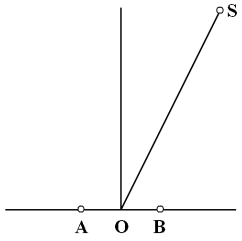

Fig. 3. Sound source localization

The goal of SSL is to estimate the angle $\angle SOB$. When the distance of the source $|OS|$, is much larger than the length of the baseline $|AB|$, the angle $\angle SOB$ can be estimated as follows [4]:

$$\angle SOB = arccos \frac{D \times v}{|AB|} \tag{18}$$

where $v = 342m/s$ is the speed of sound traveling in air.

Let the camera's optical center also be at location O. We can further convert $\angle SOB$ to object state x_c. Let β_F be the horizontal field of the view of the camera, and x_R be the horizontal resolution of the camera in pixels, we have

$$\hat{X}_t^3 = \hat{x}_t^{c,3} = \frac{x_R/2}{\tan(\beta_F/2) \cdot \tan(\angle SOB)} \tag{19}$$

The audio-based SSL tracker provides the third proposal function $q_3(X_t^3|X_{t-1}^3, Z_t^3) = N(\hat{X}_t^3, \hat{\Sigma}_t^3)$, where Σ_t^3 is the uncertainty of the SSL tracker and can be estimated from the cross correlation curve [4].

V. Verifiers used by the fuser

In the previous section, we developed three individual trackers based on the three sensors. Because we want the trackers to run in real time, simplified assumptions (e.g., Gaussianality and color constancy) are made. But as described in Section III, a sensor can have both a tracker and a verifier. As the verifier only computes the likelihood of a *given* hypothesis, more involved likelihood model can be used in the verifier, thus ensuring robust tracking.

A. The contour verifier

The contour tracker described in Section IV-A only uses the local smoothness constraint (via HMM transition probability) when detecting contours. For the contour verifier, because each hypothesis generated by the fuser is already an ellipse, it implicitly enforces both the local smoothness constraint and the prior knowledge of the elliptic shape information. We only need to check how well it matches to the detected edge in current image frame.

To calculate the contour likelihood of a given hypothesis $X_t^{(i)}$, an edge detector is applied on the normal lines of the hypothesized contour. Let $\mathbf{z}_\phi$ denote the edge detection results on line ϕ. By assuming the independence between different normal lines, the contour likelihood is:

$$p(Z_t^1|X_t^{(i)}) = \prod_{\phi=1}^{M} p(\mathbf{z}_\phi|X_t^{(i)}) \tag{20}$$

where the superscript "1" used in Z_t^1 means this is the first verifier in the paper. The term $p(\mathbf{z}_\phi|X_t^{(i)})$ is previously defined in Equation (10).

It is worth emphasizing again that in the contour tracker, it only enforces a local contour smoothness constraint. It is therefore possible that the estimated contour can be stuck on a false target, e.g., a non-elliptic object. The contour verifier, on the other hand, is much more strict and enforces the prior knowledge of the elliptic shape information. The hypothesized contour points on all normal lines therefore need to have strong edges in order to get a high likelihood score. A non-elliptic object cannot get high likelihood score because it will never match well to any elliptic hypothesis.

B. The color verifier: a discriminant model

In the color-based tracker, to achieve fast and inexpensive tracking, we made the assumption that an object's color histogram is stable and remains constant. In reality, however, the color histogram of an object changes because of lighting, shading and object motion. To handle this non-stationary nature, in the verifier we allow the object color to change but we require it to be sufficiently different from its nearby background color. That is, we use a discriminant model in the color verifier. For a given hypothesis $X_t^{(i)}$, let the object color histogram be $h^f_{X_t^{(i)}}$ and the neighboring background color histogram be $h^b_{X_t^{(i)}}$. The similarity between the two histograms can be calculated using the Bhattacharyya coefficients [20]:

$$\rho(h^f_{X_t^{(i)}}, h^b_{X_t^{(i)}}) = \sum_{l=0}^{l<Bin} \sqrt{h^f_{X_t^{(i)}}(l) \cdot h^b_{X_t^{(i)}}(l)} \tag{21}$$

where l is the index of the histogram bins. Because we use the discriminant model, the degree of difference between $h^f_{X_t^{(i)}}$ and $h^b_{X_t^{(i)}}$ furnishes the likelihood of the object. The likelihood

for hypothesis $X_t^{(i)}$ is therefore:

$$P(Z_t^2|X_t^{(i)}) = 1 - \rho(h^f_{X_t^{(i)}}, h^b_{X_t^{(i)}}) \quad (22)$$

C. The SSL verifier

In a realistic room environment, there are both ambient noise (e.g., computer fans) and room reverberation. These factors make the cross correlation curve $\hat{R}_{x_1x_2}(\tau)$ to have multiple peaks. To achieve fast tracking speed, we made a premature 0/1 decision in the SSL tracker (Section IV-C). When estimating x_t^c (Equations (17) and (18)), we only retained the time delay D but threw away the whole correlation curve. For the SSL verifier, we can afford to use more accurate likelihood model by keeping the whole correlation curve $\hat{R}_{x_1x_2}(\tau)$. Given a hypothesis $x_t^{c,(i)}$, its likelihood is defined as the ratio between its own height and the highest peak in the correlation curve $\hat{R}_{x_1x_2}(\tau)$ [16]:

$$p(Z_t^3|x_c^{(i)}) = \hat{R}_{x_1x_2}(D^{(i)})/\hat{R}_{x_1x_2}(D) \quad (23)$$

$$\hat{R}_{x_1x_2}(D^{(i)}) = \frac{|AB|}{v} cos(arctan(\frac{x_R/2}{x_c^{(i)} \cdot tan(\beta_F)})) \quad (24)$$

where Equation (23) is obtained by substituting Equation (17) into Equation (18).

By assuming independence between contour, color and audio, a combined object likelihood model is therefore:

$$p(Z_t|X_t^{(i)}) = p(Z_t^1|X_t^{(i)}) \cdot p(Z_t^2|X_t^{(i)}) \cdot p(Z_t^3|x_t^{c,(i)})$$

which is used in Equation (8) to compute the particle weights.

VI. Tracker Evaluation and Adaptation

In a non-stationary environment, object appearance can change and the background clutter (both vision and audio) can further complicate tracking. For example, when a person is turning his/her head, the color can change and causes the color-based tracker to fail. Online tracker evaluation and adaptation are therefore necessary. We can give more weights to proposals generated by the more reliable trackers. The unreliable trackers can be updated or re-initialized.

In our proposed framework, the current particle set $(X_t^{(i)}, w_t^{(i)})$ represents the estimated posterior distribution of the object state. We can estimate the reliability of each individual trackers by comparing how similar/dissimilar their proposal functions $q_k(X_t^k|X_{0:t-1}^k, Z_t^k)$ are to the estimated posterior:

$$\lambda_k = \sum_{X_t^{(i)}} \sqrt{w_t^{(i)} \cdot q_k(X_t^{(i)}|X_{0:t-1}, Z_t^k)} \quad (25)$$

This performance evaluation formula is similar to the Bhattacharyya coefficient calculation except it is based on weighted particles. The intuition behind this formula is simple: if an individual proposal function significantly overlaps with the estimated posterior, it is a good proposal function and we should trust the corresponding tracker more.

The fused tracking results can further be used to probabilistically adapt the individual trackers.

$$X_t^k = \lambda_k \hat{X}_t^k + (1 - \lambda_k) \sum_{X_t^{(i)}} w_t^{(i)} \cdot X_t^{(i)} \quad (26)$$

where $\hat{X}_t^k$ (see Section IV) is tracker k's own estimate of X_t and $\sum w_t^{(i)} \cdot X_t^{(i)}$ is the fuser's estimate of X_t. The reliability factor λ_k plays the role of an automatic regulator. If an individual tracker is reliable, the current state for that tracker depends more on its own estimate; otherwise, it depends more on the fuser's estimate.

VII. Application in Speaker Tracking

A real-time speaker tracking module based on our proposed sensor fusion framework has been designed and implemented. It has further been integrated into a distributed meeting system [1]. Our goal is to track the speaker's location so that the system can provide good views for remote participants. A fast multi-view face detector [21] detects new faces every

10 seconds to initialize the vision-based trackers. The tracked heads are marked by rectangles with different colors in Figures 4 and 5. No code optimization is done with the current system and it runs comfortably in real time on a standard Pentium 4 2.2GHz PC while tracking 5 to 6 people.

To test the robustness of the proposed algorithm, we use video sequences captured from both an office and a meeting room. The sequences simulate various tracking conditions, including appearance changes, quick movement, shape deformation, and noisy audio conditions. Sequence A, shown in Figure 4, is a cluttered office environment with 700 frames (15 frames/sec). This sequence has difficult situations for both the contour tracker and the color tracker, and we only use these two vision-based trackers to demonstrate the fusion performance. On the first row in Figure 4, the person suddenly moved his head at very fast speed. Because the contour tracker (black bounding box) restricts the contour detection to normal lines of predicted position, it loses track when the person suddenly changes his movement. But because the person's head appearance does change dramatically, the color tracker (blue bounding box) survives. On the second row, the waving hand, which has similar color to the face, greatly distracts the color tracker module. But the contour tracker succeeds by enforcing the object dynamics. The fused tracker successfully tracks the person through out the sequence by combining the two individual trackers. To better illustrate the tracking results on small images, we did not plot the bounding box for the fused tracker. But it is similar to the better one of the two individual trackers at any given time t.

Sequence B, shown in Figure 5, is a one-hour long real-life group meeting. The meeting room has computer fan noise, TV monitor noise, and has walls/whiteboards that strongly reflect sound waves. The panorama video is constructed from 5 regular IEEE 1394 video cameras, such that it has a 360 degree view of the whole meeting room [1]. In this test

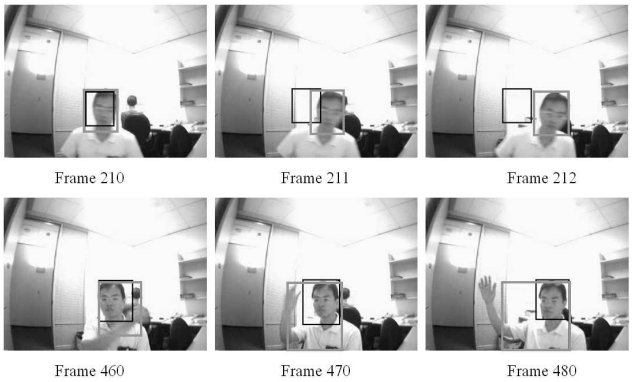

Fig. 4. Test of different proposal modules: *On the first row, the contour tracker (black bounding box) loses track when the person suddenly moves but the color tracker (blue bounding box) survives. On the second row, the color tracker module fails when the person's arm waves in front of the face, but the contour tracker succeeds.*

sequence, we use all the three trackers: the contour tracker, the color tracker and the SSL tracker. The bounding boxes with different colors are the fused tracking results from the contour tracker and color tracker. The red vertical bar is the tracking results from SSL. The green vertical bar is the fused results based on all the three trackers. The audio and vision based trackers complement each other in this speaker tracking task. Vision trackers have higher precision but are less robust while the audio tracker knows the active speaker but is with less precision. The fused tracker is robust to both vision and audio background clutter.

VIII. Conclusion

In this paper, we proposed an integrated fusion framework that combines both the bottom-up and top-down approaches to probabilistically fuse multiple sensing modalities. At the lower level, individual vision/audio trackers are designed to generate effective proposals for the fuser. At the higher level, the fuser performs reliable tracking by verifying hypotheses over multiple cues. Different from the traditional fusion algorithms, the proposed framework is a closed-loop system where the fuser and trackers dynamically exchange tracking information. To handle non-stationary situations, the fuser evaluates the performance of the individual trackers and dynamically update

Fig. 5. Speaker tracking results.

their object states. We reported robust tracking results on real-world data.

REFERENCES

[1] R. Cutler, Y. Rui, A. Gupta, J. Cadiz, I. Tashev, L. wei He, A. Colburn, Z. Zhang, Z. Liu, and S. Silverberg, "Distributed meetings: A meeting capture and broadcasting system," in *Proc. ACM Conf. on Multimedia*, 2002, pp. 123–132.

[2] Y. Rui, L. He, A. Gupta, and Q. Liu, "Building an intelligent camera management system," in *Proc. ACM Conf. on Multimedia*, 2001, pp. 2–11.

[3] http://www.polycom.com/home/.

[4] Y. Rui and D. Florencio, "Time delay estimation in the presence of correlated noise and reverberation," Microsoft Research Redmond," Technical Report MSR-TR-2003-01, 2003.

[5] Y. Chen, Y. Rui, and T. S. Huang, "Parametric contour tracking using unscented Kalman filter," in *Proc. IEEE Int'l Conf. on Image Processing*, 2002, pp. III: 613–616.

[6] K. C. Chang, C. Y. Chong, and Y. Bar-Shalom, "Joint probabilistic data association in distributed sensor networks," *IEEE Trans. Automat. Contr.*, vol. 31, no. 10, pp. 889–897, 1986.

[7] J. Sherrah and S. Gong, "Continuous global evidence-based Bayesian modality fusion for simultaneous tracking of multiple objects," in *Proc. IEEE Int'l Conf. on Computer Vision*, 2001, pp. 42–49.

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

[9] J. Vermaak, A. Blake, M. Gangnet, and P. Perez, "Sequential Monte Carlo fusion of sound and vision for speaker tracking," in *Proc. IEEE Int'l Conf. on Computer Vision*, 2001, pp. 741–746.

[10] G. Loy, L. Fletcher, N. Apostoloff, and A. Zelinsky, "An adaptive fusion architecture for target tracking," in *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, 2002, pp. 261–266.

[11] R. Merwe, A. Doucet, N. Freitas, and E. Wan, "The unscented particle filter," Cambridge University Engineering Department," Technical Report CUED/F-INFENG/TR 380, 2000.

[12] M. Isard and A. Blake, "ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework," in *Proc. European Conf. on Computer Vision*, 1998, pp. 767–781.

[13] ——, "Contour tracking by stochastic propagation of conditional density," in *Proc. European Conf. on Computer Vision*, 1996, pp. I:343–356.

[14] A. Doucet, "On sequencial simulation-based methods for Bayesian filtering," Cambridge University Engineering Department," Technical Report CUED/F-INFENG/TR 310, 1998.

[15] J. S. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, 1998.

[16] Y. Rui and Y. Chen, "Better proposal distributions: Object tracking using unscented particle filter," in *Proc. IEEE Int'l Conf. on Comput. Vis. and Patt. Recog.*, 2001, pp. II:786–794.

[17] F. Hausdorff, *Set Theory*. 3rd ed., New York: Chelsea Publishing Company, 1978.

[18] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 3, no. 1, pp. 4–15, January 1986.

[19] J. Vermaak and A. Blake, "Nonlinear filtering for speaker tracking in noisy and reverberant environments," in *Proc. IEEE Int'l Conf. Acoustic Speech Signal Processing*, 2001, pp. V:3021–3024.

[20] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Int'l Conf. on Comput. Vis. and Patt. Recog.*, 2000, pp. II 142–149.

[21] Z. Zhang, L. Zhu, S. Li, and H. Zhang, "Real-time multi-view face detection," in *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, 2002, pp. 149–154.

PLACE PHOTO HERE

Yunqiang Chen is a Researcher in the Intelligence Vision and Reasoning Group in Siemens Corporate Research. He received his B.S. degree in Electrical Engineering from Tsinghua University (China) in 1995 and his M.S. degree from National Lab of Pattern Recognition (China) in 1998. He received his Ph.D. degree from University of Illinois at Urbana-Champaign in 2002. His research interests include image/video/audio processing, medical imaging, sensor fusion, computer vision and machine learning.

PLACE PHOTO HERE

Yong Rui is a Researcher in the Communication, Collaboration and Signal Processing (CCSP) group in Microsoft Research. He received his PhD from University of Illinois at Urbana-Champaign (UIUC) in 1999. Dr. Rui's research interests include multimedia systems, real-time collaboration, distributed meetings, image/video/audio processing, computer vision and machine learning. He published one book (Exploration of Visual Data, Kluwer Academic Publishers, 2003), six book chapters, seven journal papers, and over forty referred conference papers in the above areas. Dr. Rui was co-chair of the International Workshop on Multimedia Technologies in E-Learning and Collaboration (WOMTEC), 2003. He was also on the program committees of ACM Multimedia, IEEE Computer Vision and Pattern Recognition (CVPR), IEEE Int'l Conf. on Multimedia Expo (ICME), IEEE Int'l Conf. on Image Processing (ICIP), SPIE ITCOM, and many others. Dr. Rui was on the National Science Foundation's review panel and National Academy of Engineering's Symposium of Frontiers of Engineering for outstanding researchers.