SPEAKER TRACKING USING PARTICLE FILTER SENSOR FUSION

Yunqiang Chen and Yong Rui

Microsoft Research, Redmond, WA 98052

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 different sensors. Unlike the traditional fusion algorithms, the proposed framework is a closed-loop system where the fuser and trackers coordinate their tracking information. Furthermore, the proposed framework provides a natural scheme to evaluate the performance of the individual trackers and dynamically updates their object states in non-stationary environments and hence much more robust in long-term tracking. We present a real-time speaker tracking system by fusing object contour, color and sound source localization. Robust tracking results are achieved.

1. INTRODUCTION

Distributed meetings and lectures are gaining significant attention 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 the sensor fusion part.

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 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 Gaussianity 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 is almost never true in real life, but the bottom-up scheme does not have a scheme to detect and correct the possible errors due to the simplifications.

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 person can move freely as in our scenario, a large number of hypotheses are needed in order to cover all the possible state space and 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 particle filtering in Section 2, setting the stage for the new sensor fusion framework. In Section 3, 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 4. Together, they generate good proposals for the fuser. In Section 5, we discuss the verifiers and their likelihood models based on which the hypotheses are evaluated. In Section 6, we report some promising tracking results on challenging real-world sequences. We give concluding remarks in Section 7.

2. 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, which provide a principled way for effective 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 $t$.

**Definition 1** [14]: 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:

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

The posterior distribution $p$ can be approximated by the properly weighted particles drawn from $q$ as $N \to \infty$ [14, 11]:

$$\hat{p}(X_{0:t} | Z_{1:t}) = \frac{\sum_{i=1}^{N} w_{0:t}^{(i)} \delta_{X_{0:t}^{(i)}}(dX_{0:t})}{\sum_{i=1}^{N} w_{0:t}^{(i)}}$$  \hspace{1cm} (2)

where the particle weights $w_{0:t}^{(i)}$ are calculated as:

$$w_{0:t}^{(i)} = \frac{p(Z_{1:t} | X_{0:t}^{(i)}) p(X_{0:t}^{(i)})}{q(X_{0:t} | Z_{1:t})}$$

$$w_{0:t} = \frac{\sum_{i=1}^{N} w_{0:t}^{(i)}}{\sum_{i=1}^{N} w_{0:t}^{(i)}}$$  \hspace{1cm} (3)

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 [13, 11]:

1. Current states do not depend on future observations:
   $$q(X_{0:t} | Z_{1:t}) = q(X_{0:t-1} | Z_{1:t-1}) q(X_{t} | X_{0:t-1}, Z_{1:t})$$
2. The system state is a Markov process and the observations are conditionally independent given the states:
   $$p(X_{0:t}) = p(X_{0}) \prod_{j=1}^{t} p(X_{j} | X_{j-1})$$
   $$p(Z_{1:t} | X_{0:t}) = \prod_{j=1}^{t} p(Z_{j} | X_{j})$$  \hspace{1cm} (4)

Substituting the above two equations into Eq (3), we obtain the recursive estimate for the weights:

$$w_{t}^{(i)} = \frac{p(Z_{1:t} | X_{0:t}^{(i)}) p(X_{0:t}^{(i)})}{q(X_{0:t-1}^{(i)} | Z_{1:t-1}) q(X_{0:t-1}^{(i)} | Z_{1:t-1})}$$

$$w_{t}^{(i)} = \frac{p(Z_{1:t} | X_{0:t}^{(i)}) p(X_{0:t}^{(i)})}{q(X_{0:t}^{(i)} | Z_{1:t})}$$  \hspace{1cm} (5)

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

1. **Sampling**: $N$ particles $X_{t}^{(i)}, i = 1, \ldots, N$ are sampled from the proposal function $q(X_{t} | X_{0:t-1}, Z_{1:t})$.
2. **Measurement**: Compute the particle weights using Eq (5).
3. **Output**: The weighted particles can be readily used as the tracking results. The conditional mean of $X_{t}$ can be computed using Eq (1) with $g(X_{t}) = X_{t}$, and conditional covariance of $X_{t}$ can be computed using Eq (1) with $g(X_{t}) = X_{t} X_{t}^T$.

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.

3. 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., $p(X_{t} | X_{t-1})$) used in [13]. If there is only a single sensor, an auxiliary Kalman-filter tracker is used to generate the proposal [15]. 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 it is better than single sensor and more efficient, the master-slave structure breaks the symmetry between trackers and the whole system depends on the robustness of the slave tracker. Furthermore, it does not address how to evaluate the reliability of the slave-tracker and how to adapt it when object color might change [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 4). 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 5). 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):

![Fig. 1. Fusion diagram.](image)

### 1. Tracking by individual trackers:
Using appropriate assumptions (e.g., Gaussianity and linearity), each tracker generates fast but perhaps less robust tracking results $q_k(X_{t}^{k}|X_{0:t-1}^{k}, Z_{1:t}^{k})$, where $k$ is the index for individual trackers. Different trackers are designed in Section 4.

### 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 $\lambda_{k}$ is the reliability of tracker $k$ and is estimated dynamically during tracking as explained in step 4.

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:

$$\tilde{w}_{t}^{(i)} = \frac{w_{t-1}^{(i)} \cdot \pi(Z_{t}^{(i)})}{q(X_{t}^{(i)}|X_{0:t-1}^{(i)}, Z_{1:t})}$$  \hspace{1cm} (6)

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

$$p(Z_{t}|X_{t}^{(i)}) = \prod_{k} p(Z_{t}^{k}|X_{t}^{(i)})$$  \hspace{1cm} (7)

The set of weighted particles can be readily used as the estimate of the posterior distribution (Step 3 in Section 2).

Note that each sensor has both a tracker and a verifier. The tracker tries to solve the inverse problem efficiently to guide the fuser. Small errors in the trackers can be corrected later by the fuser and verifier. 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 5). The separation of tracker and verifier strikes a good balance between efficiency and robustness.

### 4. Evaluating and adapting the trackers:
To handle the possible errors in the individual trackers due to over-simplified assumptions (e.g., color change due to shading or lighting changes), the proposal scheme also provides us a principled way for online evaluation and adaptation. Given the posterior distribution of the object state (computed in the previous step as weighted particle set $(X_{t}^{(i)}, \tilde{w}_{t}^{(i)})$), we can estimate the reliability of each individual trackers by comparing how similar/dissimilar their proposal functions $q_k(X_{t}^{k}|X_{0:t-1}^{k}, Z_{1:t}^{k})$ are to the estimated posterior:

$$\lambda_{k} = \sum_{i} \sqrt{\tilde{w}_{t}^{(i)} \cdot q_k(X_{t}^{k}|X_{0:t-1}^{k}, Z_{1:t}^{k})}$$  \hspace{1cm} (8)

This performance evaluation equation is similar to the Bhattacharyya coefficient calculation except it is based on weighted particles. The intuition behind it 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. If the reliability of a tracker is too low, we could re-initialize or adapt it based on the fused results.

The above 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 4. We discuss the verifiers in Section 5.

### 4. INDIVIDUAL TRACKERS
In this section, two vision-based trackers based on object contour and object color respectively are designed to track human
heads and an audio-based SSL tracker is used to locate the speaker. The human head is approximated as a vertical ellipse with a fixed aspect ratio of 1:2: $X_0 = [x^c_t, y^c_t, \alpha_t]$, where $(x^c_t, y^c_t)$ is the center of the ellipse, and $\alpha_t$ is the major axis of the ellipse. In Figures 3 and 4, we represent it with a bounding box. A tracker estimates its belief of the object state $X_t$ based on its own observation $z^k_t$. 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 $[x^c_t, y^c_t, \alpha_t]$, the SSL tracker only can estimates $x^c_t$.

4.1. The contour tracker

At frame $t$, edge detection is conducted along the normal lines of the predicted contour location (see Figure 2). $\sigma_{\phi} \in [1, M]$ represents the detected contour point on the $\phi$th normal line. The Kalman filter can be applied to incorporate the temporal dynamics constraint during the estimation of the contour parameters. 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:

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

where $Y_t = y^*$ is the measurement and $K_t$ is the Kalman gain. We use the constant velocity model as the object dynamics $f()$. The observation function $g()$ is derived by calculating the intersections between normal lines and the ellipse $X_t$. Because the $g()$ is nonlinear, unscented transformation is used to estimate the $K_t, X_{t[t-1]}$ and $Y_{t[t-1]}$ [5].

![Figure 2. Illustration of parametric contour tracking: The solid curve is the predicted contour. The dashed curve is the true contour. $Y_t = [x_1, \ldots, x_M]$ is the detected contour points on the normal lines. A Gaussian distributed proposal function can be formed based on the contour tracker: $q_1(X^p_t | X_{t-1}^p, Z_t^p) = N(\hat{X}^p_t, \hat{X}^p_t)$ (9) where $\hat{X}^p_t$ is the Kalman filter’s covariance matrix [8].](image)

4.2. The color tracker

Object interior (e.g. color) complements its contour in tracking. We adopt the Meanshift algorithm [16] in our system as the color tracker by assuming that the color histogram of the target $h_{obj}$ is stable. To track the object in the current frame, previous frame’s state is used as an initial guess, i.e., $X_0 = X_{t-1}^2$, 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 state $X_t^2$:

1. Let $l$ index the Meanshift iterations. Set $l = 0$.
2. Compute the color histogram at $X_t^2$: $h_{x_t^2}$. Compute the similarity between $h_{x_t}$ and the target using the Bhat-tacharya coefficient $\rho[h_{obj}, h_{x_t}]$ [16].
3. Compute the color histogram gradient and move the searching window to the new location $\hat{X}_N$ using the Meanshift analysis [16].
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 Bhat-tacharya coefficient $\rho[h_{obj}, h_{\hat{X}_N}]$.
5. If $\rho[h_{obj}, h_{\hat{X}_N}] > \rho[h_{obj}, h_{x_t}]$, goto step 6.
6. If $||X_{t} - \hat{X}_N|| < \varepsilon$, stop. Otherwise, let $l = l + 1$ and $X_t = \hat{X}_N$ and go to Step 2.

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

$$q_2(X^p_t | X_{t-1}^p, Z_t^p) = N(\hat{X}^p_t, \hat{X}^p_t)$$ (10)

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

4.3. The SSL tracker

Audio SSL can help to identify the speaker and easily locate the horizontal location $x^c_t$ of the speaker, but it is difficult for it 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 $x^c_t$, the horizontal location of the speaker. This simplifies our SSL tracker design – we only need to have two microphones to estimate $x^c_t$. 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:

$$x_1(t) = s(t - D) + h_1(t) \ast s(t) + n_1(t)$$

$$x_2(t) = s(t) + h_2(t) \ast s(t) + n_2(t)$$ (11)

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 $x_2(t)$ (i.e. $\hat{R}_{x_1x_2}(\tau)$) [4]:

$$D = \arg \max \hat{R}_{x_1x_2}(\tau)$$ (12)

Once the time delay $D$ is estimated from the above procedure, the horizontal sound source direction $x^c_t$ 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 $S$. 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]:

$$\angle{SOB} = \arccos \frac{D \times v}{|AB|}$$ (13)

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 $z_c$. Let $\beta_F$ be the horizontal field of the view of the camera, and $z_R$ be the horizontal resolution of the camera in pixels, we have:

$$\hat{X}_t^3 = \hat{X}_t^{c, 3} = \frac{\pi R/2}{\tan(\beta_F/2) - \tan(\angle{SOB})}$$ (14)

The audio-based SSL tracker provides the third proposal function $q_3(X^p_t | X_{t-1}^p, Z_t^p) = N(\hat{X}^p_t, \hat{X}^p_t)$, where $\hat{X}^3_t$ is the uncertainty of the SSL tracker and can be estimated from the cross correlation curve [4].
5. 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., Gaussianity and color constancy) are made. But as described in Section 3, 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.

5.1. The contour verifier

The contour tracker described in Section 4.1 assumes Gaussian distributed noise in contour detection which is usually not true in cluttered environments. Some strong background clutter can greatly distract the Kalman filter. The contour verifier, on the other hand, is much more strict and enforces the prior knowledge of the elliptic shape information. We also consider the color histogram of an object changes because of lighting, but threw away the whole correlation curve. For the SSL verifier, we can afford to use more accurate likelihood model by keeping the whole color 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 4.3.). When estimating \( x_t^c \) (Eqs (13) and (14)), 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 color correlation curve \( \hat{R}_{x_1x_2}(\tau) \). Given a hypothesis \( x_t^c \), its likelihood is defined as the ratio between its own height and the highest peak in the correlation curve \( \hat{R}_{x_1x_2}(\tau) \) [15]:

\[
p(\hat{Z}_t^c | x_t^c) = \hat{R}_{x_1x_2}(D(\hat{Z}_t^c)) / \hat{R}_{x_1x_2}(D) \]

where Eq (20) is obtained by substituting Eq (13) into Eq (14).

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

\[
p(\hat{Z}_t^c | x_t^c) = p(\hat{Z}_t^c | x_t^c) \cdot p(\hat{Z}_t^c | x_t^c) \cdot p(\hat{Z}_t^c | x_t^c) \]

which is used in Eq (6) to compute the particle weights.

5.2. 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 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 3, 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 (gray bounding box) survives. On the second row, the waving hand, which has similar color to the face, greatly distracts the color

\[
\rho(h_{X_t}^{(i)}, h_{X_t}^{(i)}) = \sum_{l=0}^{L} \sqrt{h_{X_t}^{(i)}}(l) \cdot h_{X_t}^{(i)}(l) \]

where \( l \) is the index of the histogram bins. Because we use the discriminant model, the degree of difference between \( h_{X_t}^{(i)} \)

and \( h_{X_t}^{(i)} \) furnishes the likelihood of the object. The likelihood for hypothesis \( x_t^{(i)} \) is therefore:

\[
P(\hat{Z}_t | x_t^{(i)}) = 1 - \rho(h_{X_t}^{(i)}, h_{X_t}^{(i)}) \]

5.3. 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 4.3.). When estimating \( x_t^c \) (Eqs (13) and (14)), 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 color correlation curve \( \hat{R}_{x_1x_2}(\tau) \). Given a hypothesis \( x_t^c \), its likelihood is defined as the ratio between its own height and the highest peak in the correlation curve \( \hat{R}_{x_1x_2}(\tau) \) [15]:

\[
p(\hat{Z}_t^c | x_t^c) = \frac{\hat{R}_{x_1x_2}(D(\hat{Z}_t^c))}{\hat{R}_{x_1x_2}(D)} \]

where Eq (20) is obtained by substituting Eq (13) into Eq (14).

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

\[
p(\hat{Z}_t^c | x_t^c) = p(\hat{Z}_t^c | x_t^c) \cdot p(\hat{Z}_t^c | x_t^c) \cdot p(\hat{Z}_t^c | x_t^c) \]

which is used in Eq (6) to compute the particle weights.

6. 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 [17] detects new faces every 10 seconds. Excel 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 3, 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 3, 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 (gray 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 throughout 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 $t$.

![Image](https://via.placeholder.com/150)

**Fig. 3.** Test of different proposal modules.

Sequence B, shown in Figure 4, 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 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 black vertical bar is the tracking results from SSL. The gray 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.

![Image](https://via.placeholder.com/150)

**Fig. 4.** Speaker tracking results.

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

### 8. REFERENCES


