Particle Filtering Algorithms for Tracking an Acoustic Source in a Reverberant Environment

Darren B. Ward, *Member, IEEE*, Eric A. Lehmann, *Student Member, IEEE*, and Robert C. Williamson, *Member, IEEE*

***Abstract*—Traditional acoustic source localization algorithms attempt to find the current location of the acoustic source using data collected at an array of sensors at the current time only. In the presence of strong multipath, these traditional algorithms often erroneously locate a multipath reflection rather than the true source location. A recently proposed approach that appears promising in overcoming this drawback of traditional algorithms, is a state-space approach using particle filtering. In this paper we formulate a general framework for tracking an acoustic source using particle filters. We discuss four specific algorithms that fit within this framework, and demonstrate their performance using both simulated reverberant data and data recorded in a moderately reverberant office room (with a measured reverberation time of 0.39 s). The results indicate that the proposed family of algorithms are able to accurately track a moving source in a moderately reverberant room.**

***Index Terms*—Acoustic signal processing, generalized cross-correlation, localization, particle filters, time-delay estimation.**

I. Introduction

THE problem of locating and tracking a wideband acoustic source in a multipath environment arises in several fields, including sonar, seismology, and speech. In this paper we are particularly interested in speech, where applications include automatic camera steering for video-conferencing, discriminating between individual talkers in multisource environments, and providing steering information for microphone arrays [1].

Traditional approaches to the above problem collect data from several microphones and use a frame of data obtained at the current time to estimate the current source location. These traditional approaches can be divided into two categories: i) time-delay estimation (TDE) methods such as the well-known generalized cross-correlation (GCC) function [2], which estimate location based on the time delay of arrival of signals at the receivers and ii) direct methods such as steered beamforming. Each method transforms the received frame of data into a function that exhibits a peak in the location corresponding to the source. We will refer to this function as the *localization function*. The practical disadvantage of these traditional approaches is that reverberation causes spurious peaks to occur in the localization function. These spurious peaks may have greater amplitude than the peak due to the true source, so that simply choosing the maximum peak to estimate the source location may not give accurate results.

A promising technique that overcomes the drawback of traditional methods is to use a state-space approach based on particle filtering (PF), as recently described in [3], [4]. The key to these new techniques is that the peak due to the true source follows a dynamical model from frame to frame, whereas there is no temporal consistency to the spurious peaks. Using a sequential Monte Carlo method, particle filters are used to recursively estimate the probability density of the unknown source location conditioned on all received data up to and including the current frame. Related work on using particle filters to track multiple moving targets can be found in [5].

In this paper, we formulate a general framework for acoustic source localization using particle filters. We assume the presence of a single acoustic source in a reverberant environment, in which the speed of wave propagation is known (and constant), and the sensor positions are also known.

The paper is organized as follows. In Section II, we formulate the source localization problem and present an overview of classical methods. These classical methods are used as a reference to compare with the tracking ability of the particle filter algorithms developed later in this paper. The general framework we propose for acoustic source localization using particle filters is described in Section III. In Section IV, we present a detailed summary of four different algorithms that fit within this framework, including those proposed in [3], [4]. Section V gives a description of the different parameters used to assess the tracking accuracy of each algorithm. In Sections VI and VII, we then present a series of experiments to test the particle filtering algorithms and compare them with the classical source localization approaches. These experiments involve both simulations based on the image method [6], and data obtained from recordings performed in a real office room.

II. Source Localization

A. Signal Model

Consider a collection of M sensors positioned arbitrarily and located in a multipath environment. Assuming a single source, the discrete-time signal received at the mth sensor (where $m = 1, \ldots, M$) is

$$x_m(k) = h_m(k) \star s(k) + n_m(k) \tag{1}$$

where $h_m(k)$ is the complete impulse response from the source to the mth sensor, $s(k)$ is the source signal, $n_m(k)$ is additive noise (assumed to be uncorrelated with the source signal and from sensor to sensor), and $\star$ denotes convolution. The impulse

Manuscript received November 2, 2002; revised April 23, 2003. This work was supported by the Australian Research Council. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Walter Kellermann.

D. B. Ward is with the Department of Electrical and Electronic Engineering, Imperial College London, SW7 2BT, U.K. (e-mail: d.ward@ic.ac.uk).

E. A. Lehmann and R. C. Williamson are with the Department of Telecommunications Engineering, Research School of Information Sciences and Engineering, The Australian National University, Canberra ACT 0200, Australia.

Digital Object Identifier 10.1109/TSA.2003.818112

response from the source to any sensor can be separated into direct path and multipath terms, giving

$$x_m(k) = \frac{1}{4\pi\|\boldsymbol{\ell}_s - \boldsymbol{\ell}_m\|} s(k-\tau_m) + s(k) \star g_m(k) + n_m(k) \quad (2)$$

where

$$\boldsymbol{\ell}_s = [\mathcal{X}_s, \mathcal{Y}_s, \mathcal{Z}_s]^T \quad (3)$$

is the source location in Cartesian coordinates, $\boldsymbol{\ell}_m$ is the sensor location, $g_m(k)$ is the component of the impulse response between the source and the mth sensor due to multipath only, and $\|\cdot\|$ denotes the vector 2-norm. The delay from the source to the mth sensor is

$$\tau_m = c^{-1}\|\boldsymbol{\ell}_s - \boldsymbol{\ell}_m\| \quad (4)$$

where c is the speed of wave propagation.

Assume that the data at each sensor are collected over a frame of L samples, and let

$$\mathbf{x}_m(t) = [x_m(tL), x_m(tL+1), \ldots, x_m(tL+L-1)] \quad (5)$$

denote the data at the mth sensor for frame t. Stack the sensor frames to form the $M \times L$ matrix

$$\mathbf{X}_t = \begin{bmatrix} \mathbf{x}_1(t) \\ \vdots \\ \mathbf{x}_M(t) \end{bmatrix} \quad (6)$$

which represents the data received at the array during time frame t. We will refer to $\mathbf{X}_t$ as the *raw data*. The problem is to estimate the current location of the source from the raw data.

B. Traditional Approach

Classical source localization algorithms attempt to determine the current source location using data obtained at the current time only. There are essentially two methods used: time-delay estimation methods (which estimate location based on the time delay of arrival of signals at the receivers) and direct methods. Each method transforms the raw data into a function that exhibits a peak in the location corresponding to the source. Here we briefly review these two methods.

1) TDE Method: Many conventional source localization algorithms partition the sensors into pairs, and attempt to find a time-delay estimate (TDE) for each pair of sensors. These TDEs can be obtained using a variety of techniques, including the adaptive eigenvalue decomposition algorithm (AEDA) [7] and the well-known generalized cross-correlation (GCC) function [2] (and its variants). These techniques act to transform the raw data into another functional form from which time delays can be estimated. Specifically, for a given time delay τ, let the P-dimensional *measurement* (where P is the number of sensor pairs) be

$$\mathbf{y}_t(\tau) = \mathbf{f}_T(\tau, \mathbf{X}_t) \quad (7)$$

where $\mathbf{f}_T(\cdot)$ is the (algorithm-dependent) function that transforms the raw data to a localization measure that exhibits a peak corresponding to the true source location. We will refer to $\mathbf{f}_T(\cdot)$ as the TDE *localization function*. As an example, we will consider the TDE localization function for the GCC technique.[1]

Let $\mathbf{x}_{p,i}(t)$ denote the data received during a given frame t at the ith sensor of the pth sensor pair, with P denoting the total number of pairs.[2] With $\mathcal{F}\{\cdot\}$ denoting the Fourier transform, $X_{p,i}(\omega) = \mathcal{F}\{\mathbf{x}_{p,i}(t)\}$ represents the frequency domain data received during time frame t at the sensors in the pth pair. With $[\mathbf{a}]_p$ denoting the pth element of the vector $\mathbf{a}$, the pth element of the TDE localization function for the GCC technique is then

$$[\mathbf{f}_T(\tau, \mathbf{X}_t)]_p = \int G_p(\omega) X_{p,1}(\omega) X^*_{p,2}(\omega) e^{j\omega\tau} d\omega \quad (8)$$

where $G_p(\omega) \in \mathbb{R}^+$ is a weighting term. One common choice for this weighting term is

$$G_p(\omega) = \frac{1}{|X_{p,1}(\omega) X^*_{p,2}(\omega)|} \quad (9)$$

which results in the well-known PHAT localization algorithm [2].

For the pth pair of sensors, the TDE is determined as

$$\hat{\tau}_p = \arg\max [\mathbf{y}_t(\tau)]_p. \quad (10)$$

Determining the TDEs for all pairs requires P one-dimensional searches over the scalar space of possible time delays.

For each pair of sensors and TDE, the locus of potential source locations in a two-dimensional setting is a line (or in three-dimensions is a hyperboloid). The source location is then estimated as the location which best fits the potential source loci across all sensor pairs. There has been a large amount of work in the literature concerning how to find this best fit (see e.g., [8]–[11]). In the simulation and experimental results of the TDE methods (GCC and AEDA) presented in Sections VI and VII, we define the source location $\hat{\boldsymbol{\ell}}_s$ as that minimizing the distance to each intersection point of the bearing lines with each other. Intersection points lying outside the room boundary are discarded, which provides an effective way of diminishing the contribution of outliers in the TDE measurements. This method is similar to that proposed in [9] and has shown a good performance for the present work compared to other variants proposed in the literature.

For the specific implementation of AEDA, we have used subsample interpolation of the delays. During each frame, we have also discarded TDEs obtained from those microphone pairs that produce a TDE that is physically not possible given the microphone spacing and sample rate used.

In practice, there are two major drawbacks with the traditional TDE approach: (i) although the true source location will usually correspond to a peak in the TDE localization function, in the presence of multipath it may not always be the global maximum; and (ii) in the presence of noise there is usually no single point at which the source loci from different sensor pairs intersect. These two drawbacks have also been addressed recently in

[1]Note that AEDA differs from GCC in that it returns a single time delay estimate, whereas GCC produces a function which has the TDE as the independent variable. The localization function for AEDA is therefore a delta function. Hence, it cannot be used with the pseudo-likelihood function described in Section III, although it could be used with the Gaussian likelihood function.

[2]If each sensor belongs to only one pair, the number of sensors M is even, and the sensor pairs are indexed in order, then $\mathbf{x}_{p,i}(t) \triangleq \mathbf{x}_{[2(p-1)+i]}(t), p = 1, \ldots, P, i = 1, 2, P = M/2$.

[12] using the notion of realizable delay vectors. Note that TDE methods are an indirect approach to source localization, since they rely on a two-stage algorithm, viz., first estimate time delays for different pairs, then combine these time delays to find the source location.

2) Direct Method: In contrast to the TDE method, direct methods attempt to estimate the source location vector without recourse to pairwise TDEs. In this case, let the scalar measurement be

$$y_t(\boldsymbol{\ell}) = f_D(\boldsymbol{\ell}, \mathbf{X}_t) \tag{11}$$

where $\boldsymbol{\ell}$ is the source location vector, and $f_D(\cdot)$ is the direct localization function. As an example of a direct localization function, consider the frequency-averaged output power of a steered beamformer (SBF) [13], [14].

Let $X_m(\omega) = \mathcal{F}\{\mathbf{x}_m(t)\}$ denote the frequency domain data received at the mth sensor during a given time frame t. For a beamformer steered to the location $\boldsymbol{\ell}$, the localization function is

$$f_D(\boldsymbol{\ell}, \mathbf{X}_t) = \int W(\omega) \left| \sum_{m=1}^{M} H_m(\boldsymbol{\ell}, \omega) X_m(\omega) \right|^2 d\omega \tag{12}$$

where the integration is computed over the frequency range of interest, $W(\omega) \in \mathbb{R}^+$ is a frequency weight, and

$$H_m(\boldsymbol{\ell}, \omega) = a_m e^{j\omega c^{-1}(\|\boldsymbol{\ell} - \boldsymbol{\ell}_m\| - d_{\text{ref}})} \tag{13}$$

is the complex-valued beamformer weighting term on the mth sensor, with $a_m \in \mathbb{R}$ the gain applied to the mth sensor signal, $\boldsymbol{\ell}_m$ the sensor location, and d_{ref} the distance to some reference point (typically chosen as the centre of the sensor array). If $a_m = 1/M, \forall\, m$, then $H_m(\omega)$ corresponds to a conventional delay-and-sum beamformer. It was shown in [15] that, if the sensor gain a_m is chosen according to the signal gain level of the source at the mth sensor, then the frequency-averaged steered beamformer output is the optimal maximum likelihood solution to locate wideband signals. It is also stated in [15] that there is no significant performance degradation if the gain is chosen as unity or is modeled by the direct path attenuation only.

The source location is estimated from the localization function as

$$\hat{\boldsymbol{\ell}} = \arg\max\ y_t(\boldsymbol{\ell}). \tag{14}$$

Determining the source location requires a single multidimensional search over the vector space of possible source locations. Although direct methods do not require the calculation of intermediate time delays, a multidimensional search over source locations is required—this is potentially computationally very demanding.

Finally, observe the similarity between the direct method (12) and the time-delay method (8). In fact, it was shown in [15] that these methods are equivalent when GCC is performed across all sensor pairs with $G_p(\omega)$ chosen according to the signal gain levels at the sensors in the pth pair, and the beamforming sensor gain a_m is chosen according to the signal gain level at the mth sensor.

III. Framework for Source Localization Using Particle Filtering

A. Development of a General Framework

In this section we formulate a general framework for source localization based on particle filtering (PF). We use a first order model of the source dynamics and define the source state at time t as

$$\boldsymbol{\alpha}_t = [\,\mathcal{X}_s, \mathcal{Y}_s, \mathcal{Z}_s, \dot{\mathcal{X}}_s, \dot{\mathcal{Y}}_s, \dot{\mathcal{Z}}_s\,]^T \tag{15}$$

where $[\mathcal{X}_s, \mathcal{Y}_s, \mathcal{Z}_s]^T$ is the true source location in Cartesian coordinates, and $[\dot{\mathcal{X}}_s, \dot{\mathcal{Y}}_s, \dot{\mathcal{Z}}_s]^T$ is the source velocity. For a given state $\boldsymbol{\alpha}$, we will denote the location vector of the state as $\boldsymbol{\ell}_{\boldsymbol{\alpha}}$.

At time t, assume that a measurement $\mathbf{y}_t$ of the unobserved state becomes available. This measurement is described by the state-space equation

$$\mathbf{y}_t = S(\boldsymbol{\alpha}_t, \mathbf{n}_1(t)) \tag{16}$$

where $S(\cdot)$ is an unknown, not necessarily linear, function of the state $\boldsymbol{\alpha}_t$ and a noise term $\mathbf{n}_1(t)$. Assume also that the state is a Markov process, which can be modeled by the state transition relation

$$\boldsymbol{\alpha}_t = T(\boldsymbol{\alpha}_{t-1}, \mathbf{n}_2(t)) \tag{17}$$

where $T(\cdot)$ is a known, not necessarily linear, function of the previous state and a noise term $\mathbf{n}_2(t)$.

Physically, the measurement $\mathbf{y}_t$ is obtained through some transformation of the raw data

$$\mathbf{y}_t(\theta) = \mathbf{f}(\theta, \mathbf{X}_t), \tag{18}$$

where we refer to θ as the *localization parameter* and $\mathbf{f}(\cdot)$ as the *localization function*. Observe that the measurements in both the TDE (7) and direct (11) methods can be described by (18). For the TDE method, $\mathbf{y}_t$ is a P-dimensional vector and θ is a scalar time delay, whereas for the direct method, $\mathbf{y}_t$ is a scalar and θ is a location vector. The common description of (18) will be used in the sequel. One should note that (16) is a state-space equation that describes the measurements as a function of the unobserved state, whereas (18) describes how the measurements are physically obtained from the raw data.

Let $\mathbf{y}_{1:t} = [\mathbf{y}_1, \ldots, \mathbf{y}_t]$ denote the concatenation of all measurements up to time t. The aim is then to recursively estimate the conditional probability density $p(\boldsymbol{\alpha}_t|\mathbf{y}_{1:t})$—the source location can be estimated as the mean or mode of this density function. Unfortunately, this posterior filtering density is usually unavailable in practice. However, assuming that the posterior density at time $t-1$ is available, then the posterior at time t can be found through prediction and update as [16]

$$p(\boldsymbol{\alpha}_t|\mathbf{y}_{1:t-1}) = \int p(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1}) p(\boldsymbol{\alpha}_{t-1}|\mathbf{y}_{1:t-1}) \times d\boldsymbol{\alpha}_{t-1} \tag{19a}$$

$$p(\boldsymbol{\alpha}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\boldsymbol{\alpha}_t) p(\boldsymbol{\alpha}_t|\mathbf{y}_{1:t-1}) \tag{19b}$$

where $p(\boldsymbol{\alpha}_t|\mathbf{y}_{1:t-1})$ is the prior, $p(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1})$ is the state transition density, and $p(\mathbf{y}_t|\boldsymbol{\alpha}_t)$ is the likelihood (or measurement density).

In general no closed-form solution exists for (19), except in the special case where $S(\cdot)$ and $T(\cdot)$ in (16) and (17) are linear, and the noise processes $\mathbf{n}_1(\cdot)$ and $\mathbf{n}_2(\cdot)$ are Gaussian. In this case the Kalman filter is the optimal solution. For the acoustic source tracking problem these functions are nonlinear, and hence, the Kalman filter cannot be used directly. A local linearization of the nonlinearity is possible, however, through the extended Kalman filter (see e.g.,[17]). An excellent tutorial overview of other suboptimal solutions to (19) is given in [18]. The approach that we will adopt in this paper is *particle filtering*. This technique is suitable for nonlinear functions and non-Gaussian noise, and approximates (19) through Monte Carlo simulation of a set of particles (representing samples of the source state) having associated discrete probability masses. As the number of particles becomes very large this set of samples and weights tends to the true distribution, and the particle filter becomes the optimal Bayesian solution to the tracking problem. The generic particle filtering algorithm is described in [18].

With this framework in place, the general algorithm that we propose for source tracking using particle filters is described in Fig. 1. This is a standard particle filtering algorithm, and only steps 2)–4) are specific to the source tracking problem. In implementing this algorithm, there are three algorithmic choices to be made: 1) what model to use for the source dynamics in Step 2); ii) what localization function to use in Step 3); and iii) how to calculate the likelihood function in Step 4).

We note that there is also choice to be made in deciding the precise implementation of the PF algorithm, although we will not deal with the many variants of PF methods in this paper (refer to [18], [19] for a review of these algorithms).

Form an initial set of particles $\{\boldsymbol{\alpha}_0^{(i)}, i = 1 : N\}$ and give them uniform weights $\{w_0^{(i)} = 1/N, i = 1 : N\}$. Then, as each new frame of data is received:

1. Resample the particles from the previous frame $\{\boldsymbol{\alpha}_{t-1}^{(i)}\}$ according to their weights $\{w_{t-1}^{(i)}\}$ to form the resampled set of particles $\{\tilde{\boldsymbol{\alpha}}_{t-1}^{(i)}, i = 1 : N\}$
2. Predict the new set of particles $\{\boldsymbol{\alpha}_t^{(i)}\}$ by propagating the resampled set $\{\tilde{\boldsymbol{\alpha}}_{t-1}^{(i)}\}$ according to the source dynamical model
3. Transform the raw data into localization measurements through application of the localization function:
$$\mathbf{y}_t(\theta) = \mathbf{f}(\theta, \mathbf{X}_t)$$
4. Form the likelihood function:
$$p(\mathbf{y}_t|\boldsymbol{\alpha}) = F(\mathbf{y}_t, \boldsymbol{\alpha})$$
5. Weight the new particles according to the likelihood function:
$$w_t^{(i)} = p(\mathbf{y}_t|\boldsymbol{\alpha}_t^{(i)})$$
and normalize so that $\sum_i w_t^{(i)} = 1$
6. Compute the current source location estimate $\hat{\boldsymbol{\ell}}_s$ as the weighted sum of the particle locations:
$$E\{\boldsymbol{\ell}_t\} = \sum_{i=1}^{N} w_t^{(i)} \boldsymbol{\ell}_{\boldsymbol{\alpha}}^{(i)}$$
7. Store the particles and their respective weights $\{\boldsymbol{\alpha}_t^{(i)}, w_t^{(i)}, i = 1 : N\}$

Fig. 1. Generic particle filtering algorithm for source tracking.

B. Source Dynamics

There are several dynamical models that could be used to represent the time-varying location of a person moving in a typical room, e.g., [20]. One that is reasonably simple but has been shown to work well in practice is the Langevin model used in [3]. In this model the source motion in each of the Cartesian coordinates is assumed to be independent. In the $\mathcal{X}$-coordinate, this motion is described as [3]:

$$\dot{\mathcal{X}}_t = a_{\mathcal{X}} \dot{\mathcal{X}}_{t-1} + b_{\mathcal{X}} F_{\mathcal{X}} \tag{20a}$$
$$\mathcal{X}_t = \mathcal{X}_{t-1} + \Delta T \dot{\mathcal{X}}_t \tag{20b}$$
$$a_{\mathcal{X}} = e^{-\beta_{\mathcal{X}} \Delta T} \tag{20c}$$
$$b_{\mathcal{X}} = v_{\mathcal{X}} \sqrt{1 - a_{\mathcal{X}}^2} \tag{20d}$$

where $F_{\mathcal{X}}$ is a normally distributed random variable, $\Delta T = L/f_s$ is the time period separating two location estimates (with L being the frame length in samples and f_s denoting the sampling frequency), $\dot{\mathcal{X}}_t$ is the source velocity, and $v_{\mathcal{X}}$ is the steady-state RMS velocity. The model parameters suggested by [3] are $\beta_{\mathcal{X}} = 10\,\mathrm{s}^{-1}$, and $v_{\mathcal{X}} = 1\,\mathrm{ms}^{-1}$; we use these parameter values in our experiments (unless stated otherwise). The dynamics and parameters for the other Cartesian dimensions are identical. In the sequel, we will use (20) to model the source dynamics.

C. Localization Function

The localization function should be chosen such that it has a maximum corresponding to the true source location. It is likely that due to multipath, the localization function may also have peaks at false locations. It is also likely that the true location may not always be the global maximum. There are two classes of possible localization function, corresponding to the two methods used for conventional source localization, viz., TDE and direct methods.

1) TDE Localization Function: In TDE localization, the sensor array is partitioned into P pairs, and the measurement is formed according to (18), where θ is a scalar time delay, and $\mathbf{y}_t$ is a $P \times 1$ vector. The localization function for the pth pair of sensors is denoted by $[\mathbf{f}(\theta, \mathbf{X}_t)]_p, p = 1, \ldots, P$. Assume that from each of these P measurements, K possible TDEs are obtained, with the kth TDE for the pth sensor pair denoted by $\hat{\theta}^{p,(k)}, p = 1, \ldots, P, k = 1, \ldots, K$. These potential TDEs would typically be obtained as the K largest local maxima of $[\mathbf{y}_t(\theta)]_p$. A practical example of an implementation of the TDE localization function is the GCC function given in (8).

Note that the localization function can be quite general, incorporating several different TDE algorithms simultaneously. For example, one of the K potential TDEs could be obtained by

AEDA with the remaining $K-1$ potential TDEs obtained from the peaks of the GCC.

2) Direct Localization Function: In direct localization, the measurement is given by (18), where θ is the location vector, and $\mathbf{y}_t$ is a scalar. Assume that from this measurement, K potential source location vectors are obtained as the largest local maxima of $\mathbf{y}_t(\theta)$, and denote the kth potential location as $\hat{\theta}^{(k)}, k = 1,\ldots,K$. A practical example of an implementation of a direct localization function is the steered beamforming function in (12).

D. Likelihood Function

For a given state $\boldsymbol{\alpha}$, the likelihood function measures the likelihood of receiving the data $\mathbf{y}_t$. The likelihood function should be chosen to reflect the fact that peaks in the localization function correspond to likely source locations. It should also reflect the fact that occasionally there may be no peak in the localization function corresponding to the true source location (such as when the source is silent). The position of the peak may also have slight errors due to noise and sensor calibration errors. We propose the following two classes of likelihood function.

1) Gaussian Likelihood: The Gaussian likelihood function we develop here is essentially identical to that proposed in [3].

If K potential locations have been obtained from the localization function, then the Gaussian likelihood function is formed by assuming that either one of these potential locations is due to the true source location corrupted by additive Gaussian noise, or none of the potential locations is due to the true source location. There are two possible ways of forming the likelihood function, depending on whether a TDE or direct localization function is used.

For a direct localization function, the likelihood function is formed as

$$F(\mathbf{y}_t,\boldsymbol{\alpha}) = \sum_{k=1}^{K} q_k \mathcal{N}(\theta_{\boldsymbol{\alpha}};\hat{\theta}^{(k)},\sigma^2) + q_0 \tag{21}$$

where $\mathcal{N}(x;m,\sigma^2)$ denotes a Gaussian distribution with mean m and variance σ^2 evaluated at x. The K potential locations obtained from the localization function are denoted by $\hat{\theta}^{(k)}, k = 1,\ldots,K$, and $\theta_{\boldsymbol{\alpha}} = \boldsymbol{\ell}_{\boldsymbol{\alpha}}$ is the localization parameter corresponding to the state.

The value of $q_0 < 1$ is the prior probability that none of the potential locations is due to the source location,[3] and $q_k < 1, k = 1,\ldots,K$ is the prior probability that the kth potential location is the true source location. Without prior knowledge of likely source locations, one would typically choose

$$q_k = \frac{1-q_0}{K}, \quad k = 1,\ldots,K. \tag{22}$$

For a TDE localization function, the likelihood function for the pth sensor pair is

$$F_p(\mathbf{y}_t,\boldsymbol{\alpha}) = \sum_{k=1}^{K} q_k \mathcal{N}(\theta^p_{\boldsymbol{\alpha}};\hat{\theta}^{p,(k)},\sigma^2) + q_0 \tag{23}$$

[3]A larger value of q_0 indicates that the true source location is often not among the K candidates. This is likely in cases where there is a high level of reverberation, or the source is often silent.

where $\hat{\theta}^{p,(k)}, p = 1,\ldots,P, k = 1,\ldots,K$ is the kth potential location obtained from the pth sensor pair, and

$$\theta^p_{\boldsymbol{\alpha}} = c^{-1}(\|\boldsymbol{\ell}_{\boldsymbol{\alpha}} - \boldsymbol{\ell}_{p,1}\| - \|\boldsymbol{\ell}_{\boldsymbol{\alpha}} - \boldsymbol{\ell}_{p,2}\|) \tag{24}$$

is the TDE corresponding to the state, with $\boldsymbol{\ell}_{p,i}$ the location of the ith sensor in the pth pair. Assuming that the measurements across sensor pairs are independent, the complete likelihood function becomes

$$F(\mathbf{y}_t,\boldsymbol{\alpha}) = \prod_{p=1}^{P} F_p(\mathbf{y}_t,\boldsymbol{\alpha}). \tag{25}$$

2) Pseudo-Likelihood: The idea behind this approach is that the localization function itself is typically a continuous function which could be used directly as the basis of a *pseudo* likelihood function. A lower bound is included to allow for the case where no peak in the localization function corresponds to the true source location. Again, there are two possible ways of forming the likelihood function, depending on whether a TDE or direct localization function is used.

For a direct localization function, the pseudo-likelihood function we propose is

$$F(\mathbf{y}_t,\boldsymbol{\alpha}) = \max\{\mathbf{y}_t(\theta_{\boldsymbol{\alpha}}),\xi_0\}^r \tag{26}$$

where $\theta_{\boldsymbol{\alpha}} = \boldsymbol{\ell}_{\boldsymbol{\alpha}}$, $\xi_0 \geq 0$, and $r \in \mathbb{R}^+$. The purpose of r is to help shape the localization function to make it more amenable to recursive estimation. The design parameter ξ_0 ensures that the pseudo-likelihood function is nonnegative (since the localization function $\mathbf{y}_t(\theta)$ can be negative), and also fulfils a role similar to that of q_0 in the Gaussian likelihood.

For a TDE localization function, the pseudo-likelihood function is

$$F(\mathbf{y}_t,\boldsymbol{\alpha}) = \prod_{p=1}^{P} F_p(\mathbf{y}_t,\boldsymbol{\alpha}) \tag{27}$$

where

$$F_p(\mathbf{y}_t,\boldsymbol{\alpha}) = \max\{[\mathbf{y}_t(\theta^p_{\boldsymbol{\alpha}})]_p,\xi_0\}^r \tag{28}$$

with $\theta^p_{\boldsymbol{\alpha}}$ given by (24), $\xi_0 \geq 0$, and $r \in \mathbb{R}^+$.

3) Discussion: The Gaussian likelihood has the advantage that it treats all peaks as being equally likely, although it requires a search over the localization function to find the peaks. The pseudo-likelihood does not require such a search, but imposes a weighting on the possible source positions proportionally to the localization function (i.e., a larger peak will be treated as a more likely source location than a smaller peak—this implicit weighting may not always be advantageous).

IV. Summary of Proposed Algorithms

The framework developed in Section III is rather general and can be implemented using a wide class of TDE or direct localization schemes. To clarify this development, here we summarize four specific PF algorithms corresponding to each of the likelihood-localization pairs that we proposed in Section III.

A. GCC Localization With Gaussian Likelihood (GCC-GL)

Organize the sensor array into P pairs. Implement the PF algorithm in Fig. 1, with steps 3) and 4) as follows. Step 3: For each sensor pair, calculate the GCC **(8)** across a set of candidate time delays. Find the K largest local maxima in the GCC function and denote the corresponding time delays as $\hat{\theta}^{p,(k)}, p = 1,\ldots,P, k = 1,\ldots,K$. Step 4: For each resampled state $\boldsymbol{\alpha}_t^{(i)}$ calculate the likelihood function $p(\mathbf{y}_t|\boldsymbol{\alpha}_t^{(i)}) = F(\mathbf{y}_t, \boldsymbol{\alpha}_t^{(i)})$ using **(23)–(25)**.

B. GCC Localization With Pseudo-Likelihood (GCC-PL)

Organize the sensor array into P pairs. Implement the PF algorithm in Fig. 1, with steps 3) and 4) as follows. Step 3: For each sensor pair, calculate the GCC **(8)** only at the time delays corresponding to the resampled states $\boldsymbol{\alpha}_t^{(i)}$, where these time delays are found using **(24)**. Step 4: For each resampled state $\boldsymbol{\alpha}_t^{(i)}$ calculate the likelihood function $p(\mathbf{y}_t|\boldsymbol{\alpha}_t^{(i)}) = F(\mathbf{y}_t, \boldsymbol{\alpha}_t^{(i)})$ using **(24), (27) and (28)**.

C. SBF Localization With Gaussian Likelihood (SBF-GL)

Implement the PF algorithm in Fig. 1, with steps 3) and 4) as follows. Step 3: Calculate the steered beamformer output power **(12)** over a set of candidate source locations. Find the K largest local maxima in the output power function and denote the corresponding location vectors as $\hat{\theta}^{(k)}, k = 1,\ldots,K$. Step 4: for each resampled state $\boldsymbol{\alpha}_t^{(i)}$ calculate the likelihood function $p(\mathbf{y}_t|\boldsymbol{\alpha}_t^{(i)}) = F(\mathbf{y}_t, \boldsymbol{\alpha}_t^{(i)})$ using **(21)**.

D. SBF Localization With Pseudo-Likelihood (SBF-PL)

Implement the PF algorithm in Fig. 1, with steps 3) and 4) as follows. Step 3: calculate the steered beamformer output power only at the set of location vectors corresponding to the resampled states $\boldsymbol{\alpha}_t^{(i)}$. Step 4: for each resampled state $\boldsymbol{\alpha}_t^{(i)}$ calculate the likelihood function $p(\mathbf{y}_t|\boldsymbol{\alpha}_t^{(i)}) = F(\mathbf{y}_t, \boldsymbol{\alpha}_t^{(i)})$ using **(26)**.

E. Discussion

Algorithm A (GCC-GL) requires the calculation of P separate GCC functions across a set of time delays. It also requires P one-dimensional searches to find the candidate TDEs. This algorithm is essentially that proposed in [3]. Note that the GCC can be implemented efficiently using the FFT, although the time delays are restricted to a specific set of values (determined by the sampling frequency and the number of points in the FFT).

Algorithm B (GCC-PL) requires calculation of P separate GCC functions only at the specific time delays corresponding to the resampled states. It is not necessary to perform any searches. Because the time delays are determined by the resampled states, however, the FFT cannot be used to calculate the GCC in this algorithm.

Algorithm C (SBF-GL) requires the calculation of a steered beamformer response over the set of all possible source locations (this set is potentially very large). Furthermore, it requires a multidimensional search to find the candidate source locations. We believe that this particular algorithm is too computationally demanding to be viable.

Algorithm D (SBF-PL) requires the calculation of the steered beamformer response only at the locations corresponding to the resampled states. No multidimensional searches are required. This algorithm was proposed in [4].

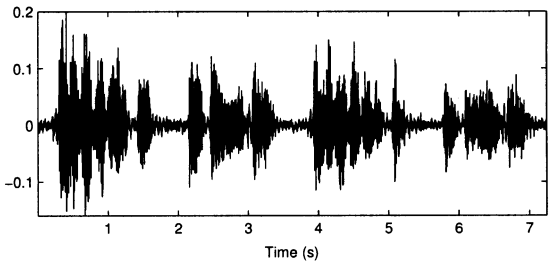

Fig. 2. Typical microphone signal simulated with the image method for $\mathrm{RT}_{60} = 132$ ms.

V. Analysis of the Tracking Accuracy

In Sections VI and VII, we will present the results from a series of experiments using simulated as well as real audio data, performed in order to determine the performance of the various algorithms proposed in Section IV together with traditional algorithms presented in Section II-B. These tests allow for a comparative assessment of the tracking ability of each method when used in a moderately reverberant and noisy environment. We first give a brief description of the different parameters used to assess the overall tracking accuracy for each algorithm simulation.

Three different parameters have been implemented in order to provide a reproducible and algorithm-independent assessment of the tracking ability. Only the first parameter (root mean square error) is applicable to the traditional localization methods described in Section II-B; the other two are based on the specific distribution of the particles for PF algorithms.

1) Root Mean Square Error (RMSE): For each frame of raw data $\mathbf{X}_t$ received from the sensors, the tracking algorithm delivers an estimate $\hat{\boldsymbol{\ell}}_s = \hat{\boldsymbol{\ell}}_s(t)$ of the current source location. The square error ε_t for time frame t is computed as:

$$\varepsilon_t = ||\boldsymbol{\ell}_s - \hat{\boldsymbol{\ell}}_s||^2. \tag{29}$$

The RMSE value then corresponds to the square root of the average value of the variable ε_t, averaged over the total number of frames in the audio sample. This parameter gives an indication about how much the source location estimate deviates from the true source position. A high RMSE value hence always reflects an inaccurate tracking ability.

2) Mean Standard Deviation (MSTD): For each time frame t, the standard deviation ς_t of a particle set around its estimate $\hat{\boldsymbol{\ell}}_s$ is defined as

$$\varsigma_t = \sqrt{\sum_{i=1}^{N} w_t^{(i)} ||\boldsymbol{\ell}_{\boldsymbol{\alpha}}^{(i)} - \hat{\boldsymbol{\ell}}_s||^2}. \tag{30}$$

Similarly to the RMSE parameter, the MSTD value corresponds to the variable ς_t averaged over the total number of frames processed by the algorithm. The MSTD parameter is an accuracy measure of the estimated source position delivered by a particle filter. A large ς_t value means that the position estimate $\hat{\boldsymbol{\ell}}_s$ results from a widely spread particle set, indicating a low level of estimation certainty.

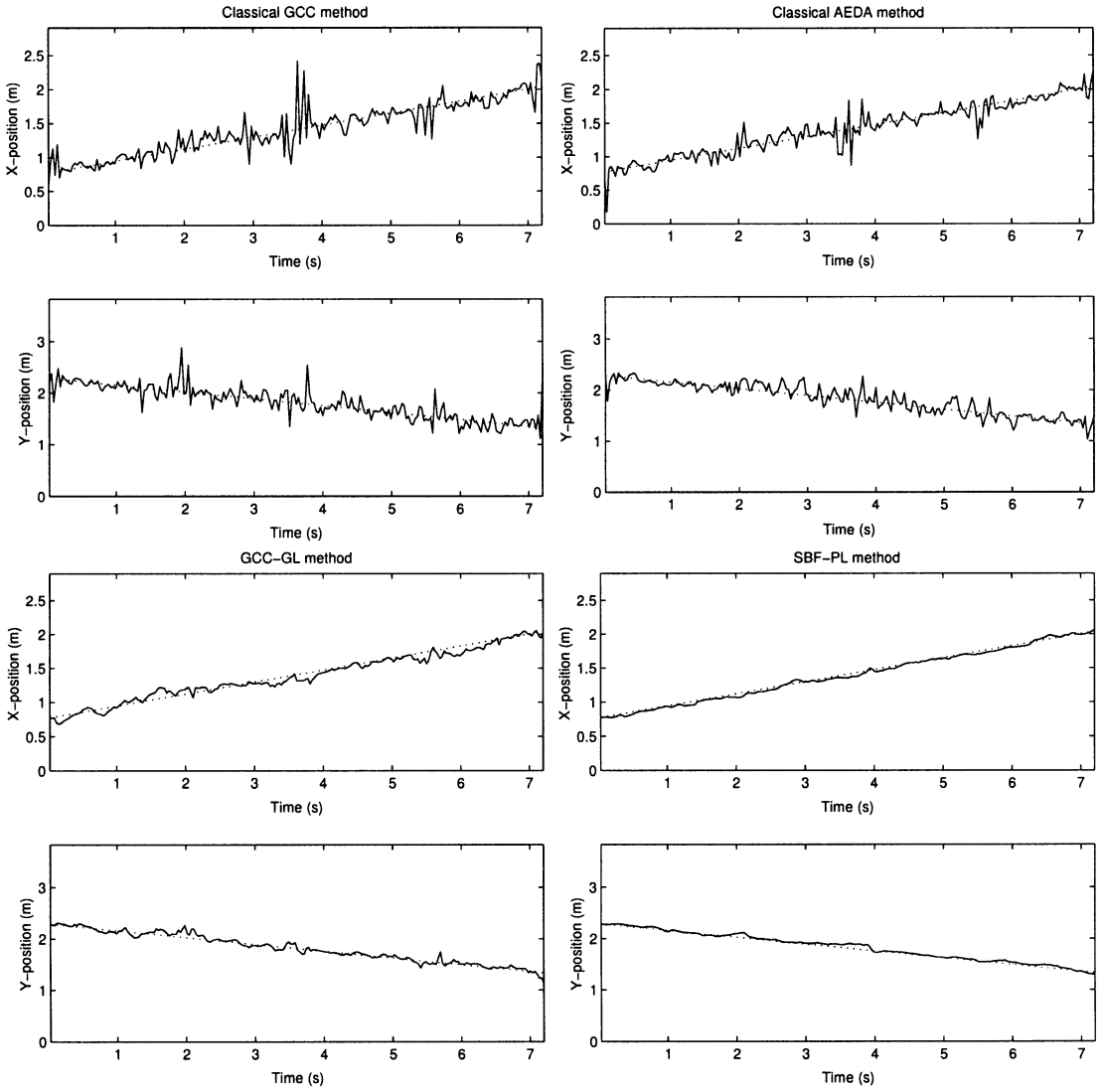

Fig. 3. Example plots showing typical tracking results for two classical and two PF-based methods. Solid lines are the algorithms' estimate of the source location, dotted lines represent true source position. The audio sample used was simulated with the image method for a reverberation time $\mathrm{RT}_{60} = 132$ ms.

3) Frame Convergence Ratio (FCR): We first define the term *convergence* as follows. For time frame t, a particle filter is said to be converging toward the true source position $\boldsymbol{\ell}_s$ if this latter lies within one standard deviation ς_t from the estimated source location $\hat{\boldsymbol{\ell}}_s$. In other words, a particle filter is convergent if the following inequality holds:

$$||\boldsymbol{\ell}_s - \hat{\boldsymbol{\ell}}_s|| \leqslant \varsigma_t + \delta \tag{31}$$

where δ accounts for the inaccuracy of the source position measurements during the recording of the audio samples. The parameter FCR is defined as the percentage of frames for which the particle filter has been found to converge, over the entire audio sample length.

It must be noted here that the FCR value depends indirectly on the MSTD parameter. If the particles are widely spread around the source location estimate, the probability of the true source lying within one standard deviation of the estimate is higher, which in turn implies a higher FCR value. Hence, a high FCR percentage may be partly resulting from a large MSTD value.

VI. Image Method Simulations

In this section, we present the tracking results obtained using synthetic audio data for the classical GCC, AEDA and SBF approaches. Results obtained from algorithms SBF-PL and GCC-GL are also shown here as generic representatives of the PF methods (these two particle filtering methods were presented in [3] and [4], respectively). We consider a two-dimensional tracking problem where the height of the source is set to be the same as the height of all the microphones.

A. Simulation Setup

For all the results presented in this section, the audio data at each sensor was obtained using the image method for simulating small-room acoustics [6] for a set of reverberation times RT_{60} ranging from 0 to 0.79 s. The simulation setup was defined to match the experimental setup used in Section VII as closely as possible: the room dimensions were set to 2.9 m $\times$ 3.83 m $\times$ 2.7 m, and 8 microphones were used in total,

the positions of which were defined as shown in Fig. 5 at the constant height of 1.464 m.

A single sample of simulated audio data has been used to obtain the results presented in this section. Fig. 5 shows the corresponding source trajectory (distance of approximately 1.6 m) defined for the image method simulations. The source signal used was the speech utterance *"Draw every outer line first, then fill in the interior"* pronounced by a male speaker and looped twice, yielding a length of about 7.2 s.

The sensors' signals were generated by splitting the source signal into 120 frames along the source's path (resulting in a frame length of about 60.4 ms). The data received at each sensor was obtained by convolving these frames of source signal with the corresponding impulse responses resulting from the image method between the source's and sensor's positions. After recombining the convolution results, random Gaussian noise was finally added to each microphone signal yielding an SNR level of about 20 dB. Fig. 2 shows a typical signal generated for microphone 1 using this setup and for a reverberation time of $\mathrm{RT}_{60} = 132$ ms.

B. Simulation Results

Each of the methods under investigation in this section was simulated for a variety of RT_{60} values and using the same setup as given above. Fig. 3 shows typical tracking results obtained for some of them for $\mathrm{RT}_{60} = 132$ ms. The tracking quality of the classical methods (especially GCC and SBF) rapidly degrades for increasing reverberation times, and plots of the tracking results for these methods become unusable for larger RT_{60} values.

Fig. 4 gives a good insight into this kind of behavior. It shows the RMSE obtained for each algorithm as a function of the RT_{60} value resulting from the image method computations. For the two PF algorithms (SBF-PL and GCC-GL), the RMSE plotted in Fig. 4 corresponds to the average RMS error resulting from 100 algorithm runs using the same audio sample and simulation setup.

C. Discussion

As clearly depicted in Figs. 3 and 4, methods based on a sequential Monte Carlo principle show a distinct improvement in tracking ability compared to more traditional source localization methods. Fig. 4 shows that for practically relevant levels of performance (i.e., for RMSE values close to zero), PF-based methods are able to deal with reverberation levels two to three times higher than classical localization methods. As has been previously reported [7], AEDA consistently gives the best results of the classical techniques.

As shown in Fig. 3, the presence of outliers in the observations computed from the raw data appear as spurious peaks in the tracking results of these classical methods. The frequency of these peaks increases dramatically as the reverberation time becomes larger, which results in a deterioration of the overall tracking ability. On the other hand, PF-based methods provide an efficient way of filtering these peaks out and hence prove to be more robust to reverberation and noise.

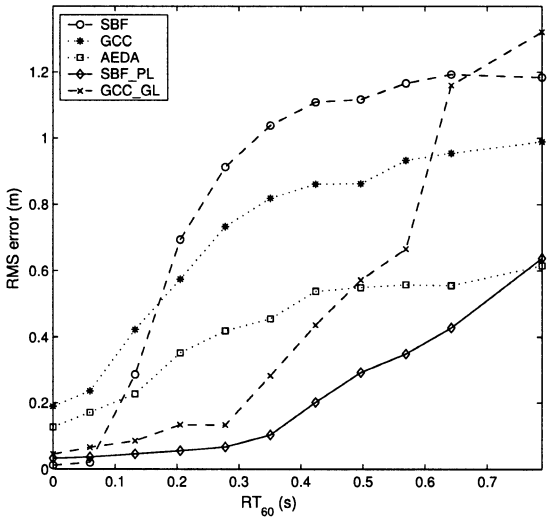

Fig. 4. Average RMS error for three classical (GCC, SBF, AEDA) and two PF-based (GCC-GL, SBF-PL) source localization methods, plotted versus reverberation time. The audio data was simulated with the image method for small room acoustics.

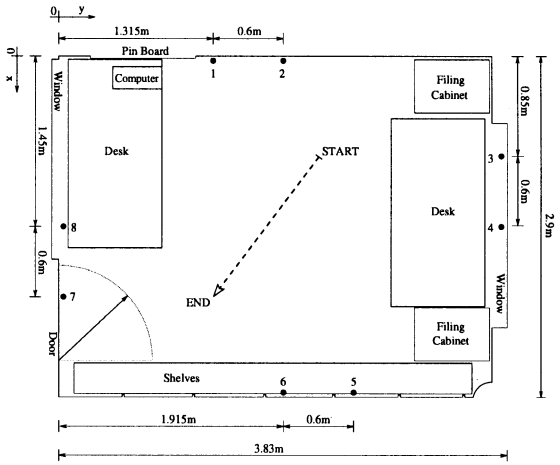

Fig. 5. Room layout with microphone positions (black circles numbered from 1 to 8, positioned at a constant height of 1.464 m), and showing a typical example of source trajectory (dashed arrow).

VII. Real Audio Experiments

In Section VI, we have described the tracking accuracy of classical as well as PF-based methods using synthetic audio data. In this section, we use real audio samples recorded in a typical office room to assess the tracking performance of these algorithms when used in a moderately reverberant environment. Again we consider a two-dimensional tracking problem.

A. Experimental Hardware Setup

The recording environment was a typical office room measuring roughly 2.9 m $\times$ 3.83 m $\times$ 2.7 m, with various encased and protruding spaces (windows, door, column, etc.). A number of office furniture objects were also present in the room during the recordings. A near to scale diagram of the room layout is presented in Fig. 5.

The level of reverberation in the room was experimentally measured by means of a loudspeaker emitting a high level white noise signal. Measuring the 60 dB decay period of the

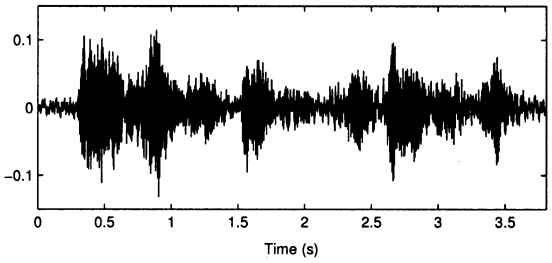

Fig. 6. Typical example of microphone signal recorded in real office room. The source signal was the sentence *"Draw every outer line first, then fill in the interior"* (taken from the TIMIT database) pronounced by a male speaker.

sound pressure level after the source signal is switched off, for a number of speaker and microphone positions, provided the frequency-averaged reverberation time $\mathrm{RT}_{60} = 0.39$ s. The level of noise in the room was comparable to typical office noise levels, including mainly a computer fan and an air-conditioning vent on the ceiling. The average SNR recorded at the microphones for the various experiments was calculated to be 9.4 dB (ranging in value from 6.8 dB for the noisiest experiment, to 16.4 dB for the best).

The experimental setup made use of a total of 8 microphones organized as one pair on each wall as depicted in Fig. 5. The moving sound source was generated in the room with a loudspeaker in upright position emitting the desired sound signal and following a predefined path at a constant height of 1.464 m (distance from the floor to the center of the speaker cone). For practical reasons, the source trajectory was always a straight line, showing a variety of lengths and orientations (mainly from one side or corner of the room to the other, within the unused floor area). Fig. 5 shows a typical example of such a source trajectory. Due to the practical method used to move the sound source in the room, a small source of error may have been introduced when monitoring the position of the speaker for the duration of the recording. The maximum deviation of the actual speaker path from the desired source trajectory was estimated to be less than 10 cm in every direction. The measurement inaccuracy parameter δ in (31) was therefore set to 0.1 m.

The audio samples used as source signals were speech utterances by male speakers with a sample length varying from 3.6 s to 7.5 s. Fig. 6 shows a typical example of sensor signal recorded with microphone 1 as the loudspeaker was moving with constant velocity across the room. The sensor signals were all sampled at 8 kHz and band-pass filtered between 300 and 3000 Hz prior to source localization processing.

B. Experimental Software Setup

1) Tuning of PF-Based Algorithms: The main objective of the simulations presented in this section is to give a comparison of the classical and PF-based methods described above with each other. In order to ensure a fair algorithm comparison, the parameters of each PF algorithm were independently tuned using a reference audio sample to achieve the best particle filter performance. This process was done empirically by running each algorithm a number of times with varying parameters until a satisfactory performance was achieved. Table I presents the parameter settings chosen for each PF algorithm. For algorithm SBF-GL, the beamformer output power was computed over a set of candidate locations distributed on a grid across the room with a uniform spacing of 0.15 m.

TABLE I
CHOSEN PARAMETER SETTINGS FOR EACH PF ALGORITHM

	N	r	q_0	ξ_0	σ	K
SBF-PL	30	3	–	0	–	–
GCC-GL	30	–	0.4	–	$1.5\mathrm{e}^{-4}$	3
SBF-GL	25	–	0.5	–	0.25	4
GCC-PL	30	0.5	–	0.01	–	–

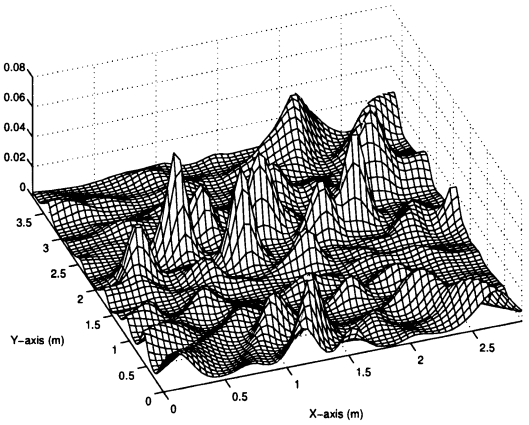

Fig. 7. Beamformer output power function used as pseudo-likelihood in algorithm SBF-PL, for one signal frame.

The PF results given in this section are obtained from the algorithms operating in tracking mode. Higher level problems (e.g., related to the detection and handling of long speech pauses, ways of initializing the particle set at simulation start, etc.) are of course important for a functional system, but we do not examine these issues in the present paper. Consequently, the particle set for each PF algorithm was initialized by placing all the particles at the start location of the sound source in the room. This way, the unpredictable effects of a uniform initial particle distribution were reduced to a negligible level. Thus, we are measuring the ability of the algorithms to track a moving source only.

Given the relatively small dimensions of the room, the variables $v_{\mathcal{X}}$ and $v_{\mathcal{Y}}$ of the source dynamics model given by (20) were set to 0.7 ms^{-1}.

2) Other Considerations: In each algorithm (classical methods included), the incoming sensor signals were split into frames of $L = 512$ samples (corresponding to a frame length of 64 ms) multiplied by a Hamming window, and the processing was carried out using a frame overlapping factor of 50%.

For the classical SBF method, the output power function of the steered beamformer was computed on a uniformly spaced grid of points across the room with a 0.02 m spacing.

C. Experimental Results

1) Example Plots: To illustrate some of the simulation results, we first present some typical plots obtained from algorithm SBF-PL using a sample of real audio data.

Fig. 7 shows an example of the function used as pseudo-likelihood plotted for one signal frame over the entire two-dimensional state-space.[4] This plot shows clearly the multihypothesis character of the observation: the peak associated with the true

[4]Note that this figure is shown for illustration purposes only: with algorithm SBF-PL, the likelihood function is evaluated *only* at the particles' positions.

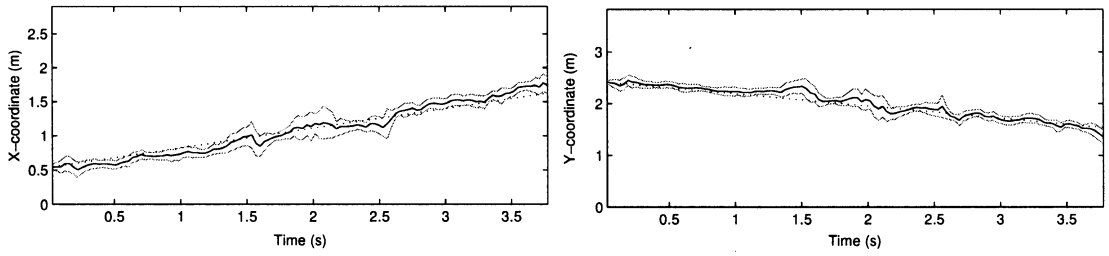

Fig. 8. Tracking results for one run of algorithm SBF-PL using real audio data, showing the true source trajectory (dotted line) and the estimated path resulting from the particle filter (solid line). Grey lines represent ± one standard deviation of the particle set from its weighted mean (estimated source position $\hat{\ell}_s$).

TABLE II
EXPERIMENTAL RESULTS: EACH OF THE 6 MAIN ROWS SHOWS THE AVERAGE PERFORMANCE MEASURES (RMSE IN M, MSTD IN M, FCR IN %) FOR A DIFFERENT SAMPLE OF REAL AUDIO DATA

	SBF-PL	SBF-GL	GCC-PL	GCC-GL	SBF	GCC	AEDA
RMSE	0.286	0.295	0.165	0.162	1.009	0.888	0.712
MSTD	0.212	0.151	0.103	0.098	–	–	–
FCR	86.1	68.8	78.2	80.3	–	–	–
RMSE	0.149	0.242	0.178	0.240	1.087	0.887	0.803
MSTD	0.182	0.164	0.109	0.109	–	–	–
FCR	97.3	79.0	82.5	67.5	–	–	–
RMSE	0.144	0.372	0.338	0.635	1.189	1.086	1.040
MSTD	0.193	0.171	0.114	0.116	–	–	–
FCR	97.7	64.4	56.3	33.9	–	–	–
RMSE	0.412	0.338	0.441	0.410	1.171	1.145	0.886
MSTD	0.219	0.176	0.113	0.111	–	–	–
FCR	74.0	81.3	42.0	49.4	–	–	–
RMSE	0.156	0.496	0.505	0.531	1.110	0.965	0.681
MSTD	0.174	0.181	0.114	0.116	–	–	–
FCR	97.8	60.7	30.8	34.1	–	–	–
RMSE	0.414	0.584	0.797	0.921	1.181	1.108	0.996
MSTD	0.247	0.177	0.118	0.116	–	–	–
FCR	79.2	55.0	24.4	19.8	–	–	–

source is located at the $(\mathcal{X}, \mathcal{Y})$ coordinate position $(0.75, 2.3)$, other peaks are clutter measurements due to reverberation.

Fig. 8 presents the tracking result in the $\mathcal{X}$ and $\mathcal{Y}$ coordinates for a 3.8 s run of algorithm SBF-PL. It demonstrates the ability of this method to accurately track the sound source across the room despite the relatively high level of reverberation. This kind of result typically yields tracking quality values of RMSE $=$ 0.114 m, MSTD $=$0.094 m and FCR $=$95%.

2) Comparative Results: The comparative results shown here have been obtained in the following manner. Each of the four PF methods under test was run 100 times with each one of 6 different samples of real audio data, implying a variety of source signals and trajectories. Since a different level of performance is usually achieved for different source signals and paths, the results obtained for each of the audio samples are given separately. Table II contains the values obtained for the performance assessment parameters averaged over the 100 real audio simulations.

As for the classical (SBF, GCC, and AEDA) results presented in this table, a single run of each algorithm has been used to generate the RMSE value for each audio sample. Contrary to PF-based methods (where the resampling and prediction steps introduce some degree of randomness), these classical methods will generate the exact same tracking results when applied twice to the same audio sample.

D. Discussion

In Table II, differences in the overall performance results from one audio sample to the other reflect a variable degree of tracking difficulty for the algorithms, resulting typically from the quality of the audio signals and the specific trajectory of the sound source. As expected, these comparative results also demonstrate the major tracking improvement of PF-based methods versus classical source localization algorithms.

When comparing PF methods only, results from Table II tend to show that algorithm SBF-PL generally works better than the other methods, yielding on average lower RMSE and higher FCR values. However, more simulations using real audio data may be required in order to fully verify this statement.

The MSTD values shown in Table II are more or less constant for each PF-based algorithm. This reflects the fact that the MSTD value is mainly resulting from the specific parameter setting chosen for each of these algorithms and that it does not strongly depend on which audio sample is used.

To give an indication of the computational complexity of each algorithm, we also measured the CPU time required to process a single audio sample.[5] Of the PF methods, both of the pseudo-likelihood algorithms (SBF-PL, and GCC-PL) took 24 s, whereas the Gaussian-likelihood algorithms took 99 s for GCC-GL, and 306 s for SBF-GL. The classical methods required 53 s for GCC, 93 s for AEDA, and several hours for SBF. As well as providing the best performance as measured in Table II, the SBF-PL algorithm is also the most computationally efficient.

[5]We have not attempted to optimize any of the algorithms used. The CPU times reported are based on Matlab implementations.

VIII. Conclusions

Carrying out acoustic source tracking in the practical environment of a moderately reverberant office room is not a trivial task. Even low levels of reverberation or background noise can rapidly become detrimental to classical TDE-based or beamforming methods. Under such adverse conditions, the use of sequential Monte Carlo methods proves to be of advantage compared to these more traditional algorithms.

In this paper, we have presented a framework for source tracking using particle filters, and discussed four specific PF-based algorithms, each of them differing from the other in the nature of the observations or in the way the measurement likelihood is computed. Results obtained from three traditional source localization methods have also been investigated and used as reference for an overall comparison of each algorithm's tracking ability.

Using synthetic audio data as well as audio samples recorded in a real office room, we have demonstrated that sequential Monte Carlo methods show a much higher degree of robustness against reverberation and background noise compared to these classical algorithms.

Acknowledgment

The authors thank K. Modrak for the practical measurements and recordings used in Section VII and A. Doucet and N. de Freitas for providing the resampling code used in the particle filter algorithm.

References

[1] M. S. Brandstein and D. B. Ward, Eds., *Microphone Arrays: Signal Processing Techniques and Applications*. Berlin: Springer-Verlag, 2001.

[2] C. H. Knapp and G. C. Carter, "The generalized correlation method of estimation of time delay," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, no. 4, pp. 320–327, 1976.

[3] J. Vermaak and A. Blake, "Nonlinear filtering for speaker tracking in noisy and reverberant environments," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP-01)*, Salt Lake City, UT, May 2001.

[4] D. B. Ward and R. C. Williamson, "Particle filter beamforming for acoustic source localization in a reverberant environment," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP-02)*, Orlando, FL, May 2002.

[5] C. Hue, J.-P. Le Cadre, and P. Pérez, "Sequential monte carlo methods for mutliple target tracking and data fusion," *IEEE Trans. Signal Processing*, vol. 50, pp. 309–325, Feb. 2002.

[6] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *J. Acoust. Soc. Amer.*, vol. 65, no. 4, pp. 943–950, 1979.

[7] J. Benesty, "Adaptive eigenvalue decomposition algorithm for passive acoustic source localization," *J. Acoust. Soc. Amer.*, vol. 107, no. 1, pp. 384–391, Jan 2000.

[8] Y. Huang, J. Benesty, G. W. Elko, and R. M. Mersereau, "Real-time passive source localization: A practical linear-correction least-squares approach," *IEEE Trans. Speech Audio Processing*, vol. 9, pp. 943–956, Nov. 2001.

[9] M. S. Brandstein, J. E. Adcock, and H. F. Silverman, "A closed-form location estimator for use with room environment microphone arrays," *IEEE Trans. Speech Audio Processing*, vol. 5, pp. 45–50, Jan. 1997.

[10] E.-E. Jan and J. Flanagan, "Sound source localization in reverberant environments using an outlier elimination algorithm," in *Proc. Int. Conf. Spoken Language Processing*, vol. 3, Philadelphia, PA, 1996, pp. 1321–1324.

[11] J. O. Smith and J. S. Abel, "Closed-form least-squares source location estimation from range-difference measurements," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 35, pp. 1661–1669, Dec. 1987.

[12] S. M. Griebel and M. S. Brandstein, "Microphone array source localization using realizable delay vectors," in *Proc. IEEE Workshop Applications Signal Processing to Audio and Acoust. (WASPAA-01)*, New Paltz, NY, Oct. 2001.

[13] J. H. Dibiase, H. F. Silverman, and M. S. Brandstein, "Robust localization in reverberant rooms," in *Microphone Arrays: Signal Processing Techniques and Applications*, M. S. Brandstein and D. B. Ward, Eds. Berlin, Germany: Springer-Verlag, 2001, ch. 8, pp. 157–180.

[14] N. Strobel, T. Meier, and R. Rabenstein, "Speaker localization using steered filtered-and-sum beamformers," in *Proc. Erlangen Workshop on Vision, Modeling, and Visualization*, Erlangen, Germany, 1999, pp. 195–202.

[15] J. C. Chen, R. E. Hudson, and K. Yao, "Maximum-likelihood source localization and unknown sensor location estimation for wideband signals in the near-field," *IEEE Trans. Signal Processing*, vol. 50, pp. 1843–1854, Aug. 2002.

[16] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian bayesian state estimation," *Proc. Inst. Elect. Eng.*, vol. 140, no. 2, pp. 107–113, Apr 1993.

[17] N. Strobel, S. Spors, and R. Rabenstein, "Joint audio-video signal processing for object localization and tracking," in *Microphone Arrays: Signal Processing Techniques and Applications*, M. S. Brandstein and D. B. Ward, Eds. Berlin, Germany: Springer-Verlag, 2001, ch. 10, pp. 203–225.

[18] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian bayesian tracking," *IEEE Trans. Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[19] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. Berlin, Germany: Springer-Verlag, 2001.

[20] M. Isard and A. Blake, "Condensation-conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, vol. 29, no. 1, pp. 5–28, 1998.

Darren B. Ward (M'96) was born in Mareeba, Australia, in 1970. He received the B.E. (Hons.) degree in electronic engineering and the B.A.S. degree in computer science from the Queensland University of Technology, Brisbane, Australia, in 1993, and the Ph.D. degree in engineering from the Australian National University, Canberra, in 1997.

From 1996 to 1998, he was a Postodoctoral Member of Technical Staff at Bell Labs, Lucent Technologies, Murray Hill, NJ. He has held academic positions at the University of New South Wales and the Australian National University, and since 2000 has been a Lecturer at Imperial College London, U.K. His research interests are in the fields of signal processing and digital communications.

Eric A. Lehmann (S'02) received the Dipl.El.-Ing. diploma from the Swiss Federal University, Zurich, in 1999 and the M.Phil. degree in electrical engineering from the Australian National University (ANU), Canberra, Australia, in 2000. He is currently pursuing the Ph.D. degree at the ANU.

His research interests include signal processing, microphone arrays, Bayesian estimation and tracking, with particular emphasis on the application of sequential Monte Carlo methods (particle filters).

Robert C. Williamson (M'90) received the B.E. degree from the Queensland Institute of Technology, Brisbane, Australia, in 1984, and the M.Eng.Sc. (1986) and Ph.D. (1990) degrees from the University of Queensland, Brisbane, all in electrical engineering.

Since 1990, he has been at the Australian National University where he is a Professor in the Research School of Information Sciences and Engineering. He is the Director of the Canberra node of National ICT Australia, a member of the Australian Research Council's Expert Advisory Committee on Mathematics, Information, and Communications Sciences and is on the editorial boards of *JMLR* and *JMLG*. His scientific interests include signal processing and machine learning.