Evolutionary optimization of dynamics models in sequential Monte Carlo target tracking

Anders M. Johansson and Eric A. Lehmann, *Member, IEEE*

***Abstract*—This article describes a new method for the online parameter optimization of various models used to represent the target dynamics in particle filters. The optimization is performed with an evolutionary strategy algorithm, by using the performance of the particle filter as a basis for the objective function. Two different approaches of forming the objective function are presented: the first assumes knowledge of the true source position during the optimization, and the second uses the position estimates from the particle filter to form an estimate of the current ground-truth data.**

The new algorithm has low computational complexity and is suitable for real-time implementation. A simple and intuitive real-world application of acoustic source localization and tracking is used to highlight the performance of the algorithm. Results show that the algorithm converges to an optimum tracker for any type of dynamics model that is capable of representing the target dynamics.

***Index Terms*—Evolutionary strategy, covariance matrix adaptation, particle filter, dynamics model, target tracking, dual estimation.**

I. Introduction

Dynamic target tracking refers to the problem of estimating the state of one or more maneuvering objects (targets) on the basis of noisy observations collected over time at one or several field sensors. Among others, the state-space concept of Bayesian filtering represents a solution of considerable importance for this type of problem definition, as demonstrated, for instance, by many existing algorithms based on the popular Kalman filtering and sequential Monte Carlo methods [1]. One particular aspect of the Bayesian filtering approach is its reliance on the definition of a suitable model describing the potential dynamics of the considered target. One challenge in dynamic target tracking is hence to appropriately choose and tune such a model in the frame of the considered application. The model and its parameters must be chosen to minimize the estimation error and the likelihood of the target being lost. The problem applies to a wide range of practical applications ranging from GPS navigation [2] and air traffic control systems [3], to acoustic source localization and tracking (ASLT) [4] and sonar systems [5]. The problem is particularly difficult in applications such as hand-held GPS receivers or military ballistic target tracking, where the system dynamics are intrinsically unknown. The difficulty increases further when ground-truth information is unavailable, since this prevents the calculation of tracking performance.

In this work, we focus on the problem of tuning model parameters to achieve optimum performance for the estimator, which is commonly known in the literature as dual estimation problem, i.e., estimation of both the target dynamics and state parameters simultaneously. The proposed solution combines an estimator based on particle filtering (PF) [1], [6] with an iterative optimization algorithm based on evolutionary strategy (ES) [7], [8] to automatically tune the parameters of the dynamics model. The ES algorithm is based on covariance matrix adaptation (CMA) [9]–[11], which has been used in previous works to optimize the parameters of various algorithms [12]–[14] and in estimating parameters for dynamic systems [15]–[18]. With the proposed method, the tuning is performed online (i.e., while the PF is tracking the target) by observing the behavior of the PF, and can be executed with or without knowledge of the true target state. The proposed algorithm is referred to as PFES and is here evaluated using an ASLT problem to give an intuitive example of how it can be applied to a real-world problem. To highlight the effectiveness of the CMA-ES algorithm in the proposed application, it is also compared to a standard (1,10)-ES [7].

To the best of our knowledge, there exist two methods for determining the parameters of dynamic models for human motion tracking, neither of which is automatic. The first is iterative empirical search, which is basically a manual version of the proposed algorithm. This method works by manually modifying the model parameters based on the performance of the tracker until an optimum set of parameters are found. This simple method is, for natural reasons, limited to models with a very low parameter count, due to its complexity and labor intensiveness. The second approach uses the laws of kinematic physics to identify model parameters based on the physical properties of a human, such as weight and maximum muscular forces. This approach works, but has a number of drawbacks. First, it requires in-depth knowledge of the target properties, which is not always available. Second, the model used for calculating the motion parameters is very complex and represents a whole research area in its own right. Third, it is not always the actual dynamics model parameters of the target that yield the best tracking performance. The reason for this is that the employed sensor system, measurement conditions and state estimation algorithm are integral parts of the target tracking problem, and hence affect the performance of the tracking algorithm. This can be illustrated by examples such as varying traction conditions, and speech pauses. It should be noted that the proposed algorithm does not estimate the true model parameters of the target, but instead produces the best model parameters *for tracking*.

This article is organized as follows. The next section gives an overview of the theory behind the different sub-algorithms, including PF and the combined PFES, and describes how the objective function used in the optimization algorithm is formed from the observations of the PF. A method on how to estimate the ground-truth of the target is presented in Section III, followed by a brief overview of two dimensional (2D) dynamics models in Section IV. The ASLT application used for the evaluation of the algorithm is presented in Section V, along with the performance results. Finally, Section VI presents some conclusions and proposes further work directions.

II. Algorithm development

The state of a single target at time instances $k = 1, 2, \dots$ can be represented by a state vector $\boldsymbol{x}_k \in \mathbb{R}^{\mathcal{N}}$ defined as

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{\ell}_k \\ \boldsymbol{s}_k \end{bmatrix}, \tag{1}$$

where $\boldsymbol{\ell}_k \in \mathbb{R}^{\mathcal{L}}$ corresponds to the target position and $\boldsymbol{s}_k \in \mathbb{R}^{\mathcal{N}-\mathcal{L}}$ to other state variables such as velocity and heading. At each time instance k, one or more sensors deliver an observation $\boldsymbol{y}_k \in \mathbb{R}^{\mathcal{P}}$ of

A. M. Johansson and E. A. Lehmann are with the Western Australian Telecommunications Research Institute (WARTI), Perth, Australia (e-mail: {ajh,ericl}@watri.org.au).

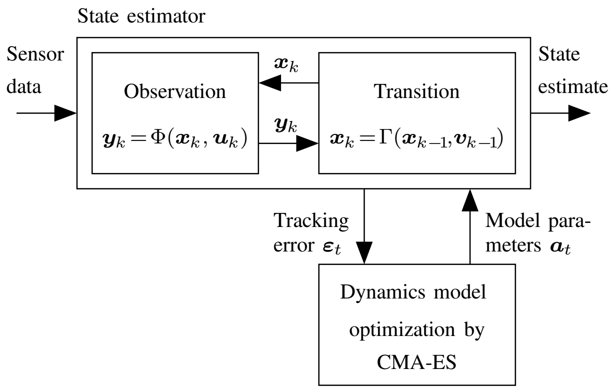
State estimator
Sensor data
Observation
$\boldsymbol{y}_k = \Phi(\boldsymbol{x}_k, \boldsymbol{u}_k)$
$\boldsymbol{x}_k$
$\boldsymbol{y}_k$
Transition
$\boldsymbol{x}_k = \Gamma(\boldsymbol{x}_{k-1}, \boldsymbol{v}_{k-1})$
State estimate
Tracking error $\boldsymbol{\varepsilon}_t$
Model parameters $\boldsymbol{a}_t$
Dynamics model optimization by CMA-ES

Fig. 1. Block diagram of PFES.

the target. The aim is to provide an estimate $\hat{\boldsymbol{x}}_k$ of the target state based on the observations.

Assuming Markovian dynamics, the system can be represented using a Bayesian filtering approach by the equations:

$$\boldsymbol{x}_k = \Gamma(\boldsymbol{x}_{k-1}, \boldsymbol{v}_{k-1}), \tag{2}$$

$$\boldsymbol{y}_k = \Phi(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{3}$$

where $\Gamma : \mathbb{R}^{\mathcal{N}} \times \mathbb{R}^{\mathcal{M}} \rightarrow \mathbb{R}^{\mathcal{N}}$ is referred to as the *transition function*, and $\Phi : \mathbb{R}^{\mathcal{N}} \times \mathbb{R}^{\mathcal{R}} \rightarrow \mathbb{R}^{\mathcal{P}}$ as the *observation function*. The transition function describes the evolution of the target state between two time instances, and the observation function provides a measurement of the target as a function of its state. The vectors $\boldsymbol{v} \in \mathbb{R}^{\mathcal{M}}$ and $\boldsymbol{u} \in \mathbb{R}^{\mathcal{R}}$ represent process and measurement noise, respectively. Note that the transition and observation functions can be nonlinear, and that the noise distributions can be non-Gaussian. The posterior probability density function (PDF) $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$, where $\boldsymbol{y}_{1:k} = \{\boldsymbol{y}_i\}_{i=1}^{k}$ is the set of all observations up to time instance k, contains all the known statistical information about the target state $\boldsymbol{x}_k$. An estimate of the target state $\hat{\boldsymbol{x}}_k$ follows from the moments of this PDF, where, for example, the target position $\hat{\boldsymbol{\ell}}_k$ can be estimated from its mean.

The transition function is determined by the physical properties of the target, the control input for the target and external forces operating on the target [19]. It is commonly formed by discretizing a continuous-time model, where the resulting discrete-time model is referred to as the *dynamics model* describing the target. The behavior of the model is controlled by $\mathcal{U}$ parameters gathered in the vector $\boldsymbol{a} \in \mathbb{R}^{\mathcal{U}}$. It is here assumed that the model itself is known or under evaluation, but that the model parameters are unknown. It is further assumed that the external forces and the control input are unknown and can be described by the noise vector $\boldsymbol{v}$.

The tracking error ε is a temporal aggregate of the difference between the true and the estimated target state. It can be minimized by optimum selection of the dynamics model parameters in $\boldsymbol{a}$. The proposed algorithm, denoted PFES, finds this optimum by observing the tracking error while slowly modifying the parameters until the optimum is reached. The optimization is performed using CMA-ES, where the objective function is calculated using the tracking error.

In the ES literature, the iteration index for the algorithm is referred to as the *generation index*, and is here denoted by t. In PFES, the generation index t is incremented at a much lower rate than the time index k of the particle filter, in order to achieve a robust estimate of the tracking error. A block-diagram of the PFES algorithm is depicted in Fig. 1. Below follows a brief review of PF and a description of the combined PFES algorithm, including a formal definition of the objective function used in the optimization.

A. Particle filtering

We wish to obtain the PDF $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ in order to estimate the target state $\hat{\boldsymbol{x}}_k$. Assuming that the PDF of the initial state $\boldsymbol{x}_0$ of the system is known, so that $p(\boldsymbol{x}_0|\boldsymbol{y}_0) = p(\boldsymbol{x}_0)$, the solution to the Bayesian filtering problem defined in (2) and (3) is determined recursively by the equations

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})\, \mathrm{d}\boldsymbol{x}_{k-1} \tag{4}$$

and

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})}, \tag{5}$$

where

$$p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}) = \int p(\boldsymbol{y}_k|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})\, \mathrm{d}\boldsymbol{x}_k. \tag{6}$$

We will here focus on the mean of the PDF $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ in order to obtain an estimate of the target position. The mean is denoted $\hat{\boldsymbol{x}}_k$, and is calculated as

$$\hat{\boldsymbol{x}}_k = \int \boldsymbol{x}_k \cdot p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})\, \mathrm{d}\boldsymbol{x}_k. \tag{7}$$

The sequential Monte Carlo approach is an approximation technique for (4) to (6). It solves the Bayesian filtering problem by representing the PDF $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ using a set of N samples of the state space (particles) with associated weights, $\{(\boldsymbol{x}_{n,k}, w_{n,k})\}_{n=1}^{N}$, where the weights are calculated based on the observations $\boldsymbol{y}_k$. Using the approach in [6], the PDF $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ and the particle state $\hat{\boldsymbol{x}}_k$ can now be approximated according to

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) \simeq \sum_{n=1}^{N} w_{n,k} \cdot \delta(\boldsymbol{x}_k - \boldsymbol{x}_{n,k}) \tag{8}$$

and

$$\hat{\boldsymbol{x}}_k \simeq \sum_{n=1}^{N} w_{n,k} \cdot \boldsymbol{x}_{n,k}, \tag{9}$$

where $\delta(\cdot)$ is the Dirac delta function. It can be shown that the variance of the weights $w_{n,k}$ among the particles gradually increases over time, which causes the accuracy of the state estimates to decrease [1], [6]. To alleviate this problem, a resampling step is introduced into the PF algorithm in order to maintain uniformity among the particles' weights and thus improve the representation of the posterior PDF. The complete algorithm is referred to as a particle filter and is summarized in Alg. 1.

B. Observation errors and performance metrics

The observations $\boldsymbol{y}_k$ given by real sensors will always be distorted due to sensor imperfections and interfering signals. Three basic types of observation errors can be identified: errors caused by noise, false observations and missing observations.

Two different performance metrics can be used to assess the effect of these different types of observation errors. The first is track loss, which occurs when the tracker fails to follow the target. Track loss typically occurs if the target makes a maneuver during a period of missing observations, or if strong noise signals or false observations occlude the true observation for an extended time period. The second performance metric is the error between the true and the estimated target position. It is used to measure the performance of the tracker during normal tracking. This metric is mostly affected by noise, but also by short durations of missing observations and by weak or

Assumption: at time $k-1$, assume that the set of particles and weights $\{(\boldsymbol{x}_{n,k-1}, w_{n,k-1})\}_{n=1}^{N}$ is a discrete representation of the posterior $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$.

Iteration: given the observation $\boldsymbol{y}_k$ obtained during the current time k, update each particle $n \in \{1, 2, \ldots, N\}$ as follows:

1. *Prediction:* propagate the particle through the transition equation, $\tilde{\boldsymbol{x}}_{n,k} = \Gamma(\boldsymbol{x}_{n,k-1}, \boldsymbol{v}_{k-1})$.
2. *Update:* assign a likelihood weight to each new particle, $\tilde{w}_{n,k} = w_{n,k-1} \cdot p(\boldsymbol{y}_k|\tilde{\boldsymbol{x}}_{n,k})$, then normalize the weights:
$$w_{n,k} = \tilde{w}_{n,k} \cdot \left(\sum_{n=1}^{N} \tilde{w}_{n,k}\right)^{-1}.$$
3. *Resampling:* compute the effective sample size,
$$\eta_{\text{EFF}} = \frac{1}{\boldsymbol{w}_k^{\text{T}} \boldsymbol{w}_k},$$
where $\boldsymbol{w}_k = [w_{1,k}, w_{2,k}, \ldots, w_{N,k}]$.
If $\eta_{\text{EFF}} \geqslant \eta_{\text{THR}}$, where η_{THR} is some pre-defined threshold, typically $0.75 \cdot N$, simply define $\boldsymbol{x}_{n,k} = \tilde{\boldsymbol{x}}_{n,k}$, $n \in \{1, 2, \ldots, N\}$. Otherwise, draw N new samples $\boldsymbol{x}_{n,k}$ from the existing set of particles $\{\tilde{\boldsymbol{x}}_{n,k}\}_{n=1}^{N}$ according to their weights $w_{n,k}$, then reset the weights to uniform values: $w_{n,k} = 1/N$, $n \in \{1, 2, \ldots, N\}$.

Result: the new set $\{(\boldsymbol{x}_{n,k}, w_{n,k})\}_{n=1}^{N}$ is approximately distributed as the posterior density $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$. An estimate of the target's location at time k can then be obtained as
$$\hat{\boldsymbol{\ell}}_k = \sum_{n=1}^{N} w_{n,k} \cdot \boldsymbol{\ell}_{n,k},$$
where $\boldsymbol{\ell}_{n,k}$ corresponds to the position information contained in the n-th particle vector: $\boldsymbol{x}_{n,k} = \left[\boldsymbol{\ell}_{n,k}^{\text{T}}, \boldsymbol{s}_{n,k}^{\text{T}}\right]^{\text{T}}$. Further, a measure of the confidence level in the PF estimate $\hat{\boldsymbol{\ell}}_k$ can be computed as the standard deviation ς_k of the particle set:
$$\varsigma_k = \sqrt{\sum_{n=1}^{N} w_{n,k} \cdot \|\boldsymbol{\ell}_{n,k} - \hat{\boldsymbol{\ell}}_k\|^2}.$$

Alg. 1. Bootstrap PF algorithm.

intermittent false observations. Improper modeling of target dynamics will also lead to increased track loss and tracking error.

The squared distance ε_k between the true and the estimated source positions $\boldsymbol{\ell}_k$ and $\hat{\boldsymbol{\ell}}_k$ at observation index k is defined as
$$\varepsilon_k = \|\boldsymbol{\ell}_k - \hat{\boldsymbol{\ell}}_k\|^2. \tag{10}$$
Let ρ_k denote the instantaneous track state at observation index k, where $\rho_k \equiv 0$ indicates a track loss and $\rho_k \equiv 1$ the alternative. Using a two state model, we define the track state as
$$\rho_k = \begin{cases} 0 & \text{if } \sqrt{\varepsilon_k} > \varepsilon_{\text{U}} \text{ or } \rho_{k-1} = 0 \text{ and } \sqrt{\varepsilon_k} > \varepsilon_{\text{L}}, \\ 1 & \text{otherwise,} \end{cases} \tag{11}$$
with $\varepsilon_{\text{U}} > \varepsilon_{\text{L}}$. Thus if the distance between the estimated and the true source position is above ε_{U}, we assume that the tracker has lost track of the target, and if it goes below ε_{L}, that tracking has resumed.

The squared distance ε_k and the track state ρ_k give an indication of the instantaneous performance of a tracking algorithm. Averaging the measurements over time yields two distinct figures of merit that can be used to assess the effectiveness of the tracker. The first is the track loss percentage (TLP), which indicates how often the tracking algorithm loses track of a target. It is denoted $\overline{\rho}$ and is calculated from a sequence of K track state measurements according to
$$\overline{\rho} = 100 \cdot \frac{\sum_{k=1}^{K}(1-\rho_k)}{K}. \tag{12}$$
The second measurement is the root-mean-squared error (RMSE), denoted by $\overline{\varepsilon}$, for the tracker. The RMSE is only defined using time samples during which the tracker is tracking properly, i.e., when $\rho_k \equiv 1$, and is calculated as
$$\overline{\varepsilon} = \sqrt{\frac{1}{\sum_{k=1}^{K} \rho_k} \boldsymbol{\rho}^{\text{T}} \cdot \boldsymbol{\varepsilon}}, \tag{13}$$
where $\boldsymbol{\rho} = [\rho_1, \rho_2, \ldots, \rho_K]$ and $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_K]$.

C. Objective function definition

The problem of finding the optimum dynamics model parameters can be formulated as the optimization problem
$$\varphi_{\text{OPT}} = \min_{\boldsymbol{a}} \Psi(\boldsymbol{a}), \tag{14}$$
where $\Psi : \mathbb{R}^{\mathcal{U}} \rightarrow \mathbb{R}$ is the objective function, and $\varphi_{\text{OPT}} \in \mathbb{R}$ is the optimum objective function value. Evolution strategies are a type of stochastic optimization algorithms that can solve the above problem based only on the value of the objective function, even when it is time-varying and has been corrupted by noise [13], [20], which makes it suitable for the application presented here. Furthermore, ES integrates with the particle filter in a way which makes it possible to perform the optimization while the particle filter is tracking the target.

The innovative approach used here to form the objective function is to divide the particle set into L equally sized groups, each with their own parameter vector $\boldsymbol{a}_l$. The tracking performance for each group corresponds to how well the parameter set $\boldsymbol{a}_l$ fits the target dynamics, and is used as objective function for the optimization algorithm. Thus, the objective function is here formed, on a smaller scale, using the same figure of merit as that used in order to assess the performance for the whole estimator, namely the tracking error ε_k. By using this approach, the optimization will automatically be performed both with respect to the RMSE and the TLP since the tracking error will in general assume a high value both during poor tracking and during track loss.

This approach is crucial for a successful optimization of the model parameters compared to, for instance, an implementation where each particle is allowed to have its own independent parameter vector. The reason behind this is that the specific interaction between particles at any given time step within a PF algorithm cannot be emulated by considering the evolution of one single particle over time. With the proposed approach, each subgroup of particles effectively evolves according to a specific transition function, while the model optimization and the PF update processes take place across the different subgroups.

With $P = N/L$ particles in each group, the particle subgroups $l = 1, 2, \ldots, L$, are effectively defined as $\{(\boldsymbol{x}_{n,k}, w_{n,k})\}_{n=P(l-1)+1}^{Pl}$. Using only the particles in the l-th subgroup, a source position estimate then follows as
$$\hat{\boldsymbol{\ell}}_{k,l} = \sum_{n=P(l-1)+1}^{Pl} \frac{w_{n,k} \cdot \boldsymbol{\ell}_{n,k}}{\sum_{n=P(l-1)+1}^{Pl} w_{n,k}}, \tag{15}$$
where the numerator is necessary to normalize the weights within each subgroup. To minimize both the TLP and the RMSE, the objective function is calculated from the estimation error between the position of each subgroup and the true target position. Thus, the objective function value for group l in generation t is calculated by

averaging the squared distance between the estimated and true source position for the group over time, according to

$$\varphi_{l,t} = \Psi(\boldsymbol{a}_{l,t}) \tag{16}$$

$$= \sum_{i=G(t-1)+1}^{Gt} \|\boldsymbol{\ell}_i - \hat{\boldsymbol{\ell}}_{i,l}\|^2. \tag{17}$$

The relation between the two time indices k and t is $t \cdot G = k$, with the integer constant $G \gg 1$ corresponding to the number of position estimates used to calculate one objective function value. The parameter vectors and the objective function values form a set of pairs $\{(\boldsymbol{a}_{l,t}, \varphi_{l,t})\}_{l=1}^{L}$ that can be used by the ES algorithm.

The noise contained in the objective function has several interesting properties, adding to the complexity of the optimization problem at hand. Three distinctive sources of noise affecting the objective function can be identified. The first is measurement noise originating from the calculation of the source position $\hat{\boldsymbol{\ell}}_{k,l}$. Since the noise originates from the aggregation in (17), of measurement errors, it can be assumed to be zero mean Gaussian with the standard deviation σ_{M}, and is here for simplicity assumed to be temporally uncorrelated. The second source of noise originates from varying dynamics of the target trajectory, where some parts of the trajectory can contain a number of slow turns while others can contain sections of very rapid linear motion, for instance. This process shifts the location of the optima as a function of time, and can be modeled as time correlated noise added to the vector $\boldsymbol{a}$. A simple, but yet effective model for this type of noise is zero mean white Gaussian noise with standard deviation σ_{T}, which is exponentially averaged with forgetting factor $\beta_{\text{T}} < 1$. The last noise process is mainly present during early convergence of the optimization algorithm and is caused by the initial poor choice of model parameters. It causes the particle filter to track the target poorly and does therefore result in an increased measurement error. By repeating the approach used for the measurement noise originating from the calculation of the source position, it can be modeled as zero mean white Gaussian noise which is added to the objective function. To properly represent the dependence on the poor choice of model parameters, the noise power is defined to be proportional to the distance to optimum, and is hence defined as $\sigma_{\text{C}}\|\boldsymbol{a}_k - \boldsymbol{a}_{\text{OPT}}\|^2$.

The above noise models can be combined to form a model of the objective function:

$$\Psi(\boldsymbol{a}_k) = \sigma_{\text{M}} z_{\text{M},k} + \sigma_{\text{C}}\|\boldsymbol{a}_k - \boldsymbol{a}_{\text{OPT}}\|^2 z_{\text{C},k} + \Pi(\boldsymbol{a}_k + \boldsymbol{v}_k), \tag{18}$$

where

$$\boldsymbol{v}_k = (1-\beta_{\text{T}})\boldsymbol{v}_{k-1} + \beta_{\text{T}}\sigma_{\text{T}}\boldsymbol{z}_{\text{T},k}, \tag{19}$$

z_{M} and z_{C} are $(0,1)$-normal distributed random numbers and $\boldsymbol{z}_{\text{T}}$ is a $(\mathbf{0},\mathbf{I})$-normal distributed random vector with uncorrelated elements. Furthermore, the function $\Pi : \mathbb{R}^{\mathcal{U}} \to \mathbb{R}$ is a noise-free version of the objective function, the vector $\mathbf{0} \in \mathbb{R}^{\mathcal{U}}$ is the zero vector and $\mathbf{I} \in \mathbb{R}^{\mathcal{U}\times\mathcal{U}}$ is the identity matrix. In the evaluation, it will be shown that the noise-free objective function Π is a smooth convex function in the considered application.

D. PFES Algorithm

Evolution strategies were pioneered by Rechenberg and Schwefel in 1963 [21]. Their basic $(1+1)$-ES algorithm uses two individuals, one parent and one child, identified by the vectors $\boldsymbol{a}_{\text{P},t}$ and $\boldsymbol{a}_{\text{C},t}$ respectively, where the child is generated by adding $(\mathbf{0},\mathbf{I})$-normal distributed noise $\boldsymbol{z}_t$ to the parent:

$$\boldsymbol{a}_{\text{C},t} = \boldsymbol{a}_{\text{P},t} + \boldsymbol{z}_t. \tag{20}$$

The objective function values for the parent and the child are then calculated and compared, and the individual corresponding to the lowest value is used as parent in the next generation. The method can be generalized to (M,L)-ES methods, which use a larger population with M parents and L offspring, where the whole population is replaced in every generation. A number of steps can be taken to improve the convergence rate and quality of the solution [7], as listed below.

1) **Several parents per child**. ES algorithms denoted $(M_{\boldsymbol{q}}, L)$ calculate a "super parent" $\overline{\boldsymbol{a}}_t$ for all the children in every generation as a weighted average of the M best members of the population:
$$\overline{\boldsymbol{a}}_t = \sum_{l=1}^{M} \tilde{\boldsymbol{a}}_{l,t} \cdot q_l,$$
where $\boldsymbol{q} = [q_1, q_2, \ldots q_M]$ is a fixed weighting vector, and the members $\tilde{\boldsymbol{a}}_{l,t}$ correspond to the parent individuals $\boldsymbol{a}_{l,t}$ sorted with respect to their corresponding objective function value in ascending order. Uniform weighting was used in early works [22], [23]. However, this approach has been superseded by non-uniform weighting [9], for which $q_l > q_{l+1}, l \in \{1,2,\ldots,M-1\}$.
2) **Introduction of step size**. The amplitude of the noise in (20) is controlled by a parameter σ_t, whose value is decreased as a function of t [24]:
$$\boldsymbol{a}_{l,t+1} = \overline{\boldsymbol{a}}_t + \sigma_t \cdot \boldsymbol{z}_{l,t+1} \qquad \text{for } l = 1,2,\ldots,L.$$
3) **Individual step sizes for each dimension**. The step size is controlled individually for each element in $\boldsymbol{a}$ depending on the shape of the objective function [25]:
$$\boldsymbol{a}_{l,t+1} = \overline{\boldsymbol{a}}_t + \sigma_t \boldsymbol{D}_t \cdot \boldsymbol{z}_{l,t+1} \qquad \text{for } l = 1,2,\ldots,L,$$
where $\boldsymbol{D}_t$ is a diagonal matrix with the relative step sizes for each element of $\boldsymbol{a}$ on its diagonal.
4) **Adaptive step-size update**. The above mentioned step sizes are automatically updated by analyzing the statistical properties of the noise used to generate the parents in previous generations [9], [26].
5) **Correlation between elements of $\boldsymbol{a}$**. The amplitude of the noise added in each element of $\boldsymbol{a}$ is calculated by incorporating the dependence between the elements of the vector [9]:
$$\boldsymbol{a}_{l,t+1} = \overline{\boldsymbol{a}}_t + \sigma_t \boldsymbol{Q}_t \boldsymbol{D}_t \cdot \boldsymbol{z}_{l,t+1} \qquad \text{for } l = 1,2,\ldots,L,$$
where each element of the matrix $\boldsymbol{Q}_t$ contains the relative interdependence.

The CMA-ES algorithm developed by Hansen and Ostermeier [9] incorporates all the above improvements. In the algorithm, the individual step-size parameters in $\boldsymbol{D}_t$ and the parameter correlations in $\boldsymbol{Q}_t$ are calculated from a time average of the covariance matrices of the noise used to generate the parents. The step-size adaptation is performed by analyzing the so-called evolution path. The same approach is used to speed up the adaptation of $\boldsymbol{D}_t$ and $\boldsymbol{Q}_t$ (see [9] for details).

As mentioned in Section II-C, different parts of the target trajectory may have different dynamics. The result is that $\overline{\boldsymbol{a}}_t$ will vary between generations, not only because the algorithm is converging, but also due to the dynamics of the trajectory. To avoid degenerate solutions, we propose a solution, similar to the one suggested in [27], that prevents a rapid change of $\overline{\boldsymbol{a}}_t$ by using an exponential average as follows:

$$\overline{\boldsymbol{a}}_t = (1-\beta) \cdot \overline{\boldsymbol{a}}_{t-1} + \beta \sum_{l=1}^{M} \tilde{\boldsymbol{a}}_{l,t} \cdot q_l, \tag{21}$$

where β is the forgetting factor. It is important to note that this change does not affect the convergence rate since the noise vector $\overline{\boldsymbol{z}}_t$

TABLE I
CONSTANTS USED BY THE PFES ALGORITHM.

Constant	Value	Purpose
$\mathcal{U}$	$\mathcal{U} \in \{1, 2, 4\}$	Dimensionality of problem, i.e., length of vector $\boldsymbol{a}$
M	$4 + \lfloor 3 \ln(\mathcal{U}) \rfloor$	Number of parents per generation
L	$2M$	Number of children per generation
q_i	$\ln\left(\frac{M+1}{2}\right) - \ln(i)$	Weight vector, $i \in \{1, 2, \ldots, M\}$
γ_C	$\frac{4}{\mathcal{U}+8}$	Cumulation time for $\boldsymbol{p}_\mathrm{C}$
γ_CU	$\sqrt{\gamma_\mathrm{C}(2-\gamma_\mathrm{C})}$	Variance normalization [9]
γ_Q	$\frac{\sum_{l=1}^{M} q_l}{\lVert \boldsymbol{q} \rVert}$	Variance modfier [9]
γ_COV	$\frac{2\gamma_\mathrm{Q}^2 - 1}{(N+30)^2 + 2\gamma_\mathrm{Q}^2}$	Change rate for $\boldsymbol{C}$
γ_σ	$\frac{4}{\mathcal{U}+4}$	Cumulation time for $\boldsymbol{p}_\sigma$
$\gamma_{\sigma\mathrm{U}}$	$\sqrt{\gamma_\sigma(2-\gamma_\sigma)}$	Variance normalization [9]
τ_σ	$1 + 20\gamma_\sigma^{-1}$	Damping factor change in σ
$\hat{\chi}_\mathrm{M}$	$\mathbb{E}\{\lVert \mathcal{N}(\mathbf{0}, \mathbf{I}) \rVert\}$	Expected length of a $(\mathbf{0}, \mathbf{I})$-normally distributed random vector

(see Alg. 2) remains unchanged. Further immunity to sudden changes in the objective function is achieved by doubling the size of the population compared to the values proposed in [9]. This also ensures that a larger set of dynamics parameters is available to the particle filter during early adaptation. This reduces the risk of track loss and does therefore improve the convergence rate.

The various constants used in the resulting PFES algorithm are summarized in Table I, while the algorithm itself is presented in Alg. 2. The values of τ_σ and γ_COV are chosen to slow down convergence in order to expose the algorithm to a wider variety of trajectory dynamics before it converges. This is necessary since too rapid a convergence will yield a tracker which is optimized on only a small set of the potential trajectory dynamics, and is hence sub-optimal in a global sense.

E. Discussion

An interesting observation resulting from the implementation of the proposed algorithm is how similar the PF and the ES algorithms are, as explained below.

1) **Data set**. Both algorithms have a set of vectors and weights, $\{(\boldsymbol{x}_n, w_n)\}_{n=1}^N$ and $\{(\boldsymbol{a}_l, \varphi_l)\}_{l=1}^L$, for PF and ES respectively.
2) **Propagation/mutation**. During the PF propagation, a new set of vectors is generated by transforming the old set and by adding noise. This corresponds to the mutation in the ES algorithm where colored noise is added to the transformed parent set to generate the offspring.
3) **Likelihood/objective function evaluation**. The PF evaluates the "fitness" w_n of each state vector through an observation which is aggregated over time between resampling steps. A similar operation is performed in the ES in the objective function evaluation when φ_l is calculated.
4) **Resampling/selection**. In the ES, selection is based on the objective function value with crossover between parents. The same is done in the PF during resampling, where particles with high weight are duplicated.

This comparison shows that the most significant difference between the two methods is that the PF propagates particles between optimization steps, while the ES does not. This indicates that it should be possible to use a lot of ideas from both the ES and PF fields to come up with new improved algorithms, and progress has already been done in that respect [28]–[30].

Initialization: initialize all dynamic variables according to $\boldsymbol{D}_1 = \mathbf{I}$, $\boldsymbol{Q}_1 = \mathbf{I}$, $\boldsymbol{p}_{\sigma,1} = \mathbf{1}$, $\boldsymbol{p}_{\mathrm{C},1} = \mathbf{1}$ and $\sigma_1 = 0.5$, where $\mathbf{I}$ denotes the identity matrix and $\mathbf{1}$ the unity vector. The values of $\boldsymbol{a}_{l,0}, l \in \{1, 2, \ldots, L\}$, are randomly distributed over the objective function space.

Iteration: for each generation t, a new set of offspring is calculated according to

1. *Selection:* the objective function for the current generation is evaluated as
$$\varphi_{l,t} = \sum_{i=G(t-1)+1}^{Gt} \lVert \boldsymbol{\ell}_i - \hat{\boldsymbol{\ell}}_{i,l} \rVert^2 \quad \text{for } l \in \{1, 2, \ldots, L\},$$
where $\hat{\boldsymbol{\ell}}_{i,l}$ is the position estimate from the l-th subgroup of particles in the PF, computed via (15). The individuals and their corresponding noise vectors are then sorted in ascending order with respect to their objective function values:
$$\{(\tilde{\boldsymbol{a}}_{l,t}, \tilde{\boldsymbol{z}}_{l,t})\}_{l=1}^L = \mathrm{sort}\Big(\{(\boldsymbol{a}_{l,t}, \boldsymbol{z}_{l,t})\}_{l=1}^L, \{\varphi_{l,t}\}_{l=1}^L\Big),$$
where $\mathrm{sort}(\boldsymbol{x}, \boldsymbol{y})$ sorts both the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in the same order using the values found in vector $\boldsymbol{y}$ to determine the sort order.
2. *Combination:* calculate the weighted averages $\overline{\boldsymbol{a}}_t$ and $\overline{\boldsymbol{z}}_t$ from the M best individuals and their corresponding noise vectors respectively as[1]
$$\overline{\boldsymbol{a}}_t = (1-\beta) \cdot \overline{\boldsymbol{a}}_{t-1} + \beta \sum_{l=1}^{M} \tilde{\boldsymbol{a}}_{l,t} \cdot q_l,$$
$$\overline{\boldsymbol{z}}_t = \sum_{l=1}^{M} \tilde{\boldsymbol{z}}_{l,t} \cdot q_l.$$
3. *Covariance matrix update:* the evolution path $\boldsymbol{p}_\mathrm{C}$ for the covariance matrix is updated and is used to update the noise covariance matrix $\boldsymbol{C}$:
$$\boldsymbol{p}_{\mathrm{C},t+1} = (1-\gamma_\mathrm{C}) \cdot \boldsymbol{p}_{\mathrm{C},t} + \gamma_\mathrm{CU} \cdot \gamma_\mathrm{Q} \cdot \boldsymbol{Q}_t \boldsymbol{D}_t \overline{\boldsymbol{z}}_t,$$
$$\boldsymbol{C}_{t+1} = (1-\gamma_\mathrm{cov}) \cdot \boldsymbol{C}_t + \gamma_\mathrm{cov} \cdot \boldsymbol{p}_{\mathrm{C},t+1} (\boldsymbol{p}_{\mathrm{C},t+1})^\mathrm{T}.$$
4. *Step-size update:* the evolution path $\boldsymbol{p}_\sigma$ for the step-size parameter σ is updated and is used to update the step size according to
$$\boldsymbol{p}_{\sigma,t+1} = (1-\gamma_\sigma) \cdot \boldsymbol{p}_{\sigma,t} + \gamma_{\sigma\mathrm{U}} \cdot \gamma_\mathrm{Q} \cdot \boldsymbol{Q}_t \overline{\boldsymbol{z}}_t,$$
$$\sigma_{t+1} = \sigma_t \cdot \exp\left(\frac{1}{\tau_\sigma} \cdot \frac{\lVert \boldsymbol{p}_{\sigma,t+1} \rVert - \hat{\chi}_\mathrm{M}}{\hat{\chi}_\mathrm{M}}\right).$$
5. *Offspring generation:* new $\boldsymbol{D}_{t+1}$ and $\boldsymbol{Q}_{t+1}$ matrices are calculated through singular value decomposition of $\boldsymbol{C}_{t+1}$, and are used along with the updated step size and the "super parent" to calculate a new set of offspring:
$$\boldsymbol{a}_{l,t+1} = \overline{\boldsymbol{a}}_t + \sigma_{t+1} \boldsymbol{Q}_{t+1} \boldsymbol{D}_{t+1} \cdot \boldsymbol{z}_{l,t+1},$$
for $l \in \{1, 2, \ldots, L\}$, and where $\boldsymbol{z}_{l,t+1}$ is a $(\mathbf{0}, \mathbf{I})$-normal distributed random noise vector.

Alg. 2. The PFES algorithm.

[1]The forgetting factor β is set to 1 for $t \equiv 1$ to initialize $\overline{\boldsymbol{a}}_t$.

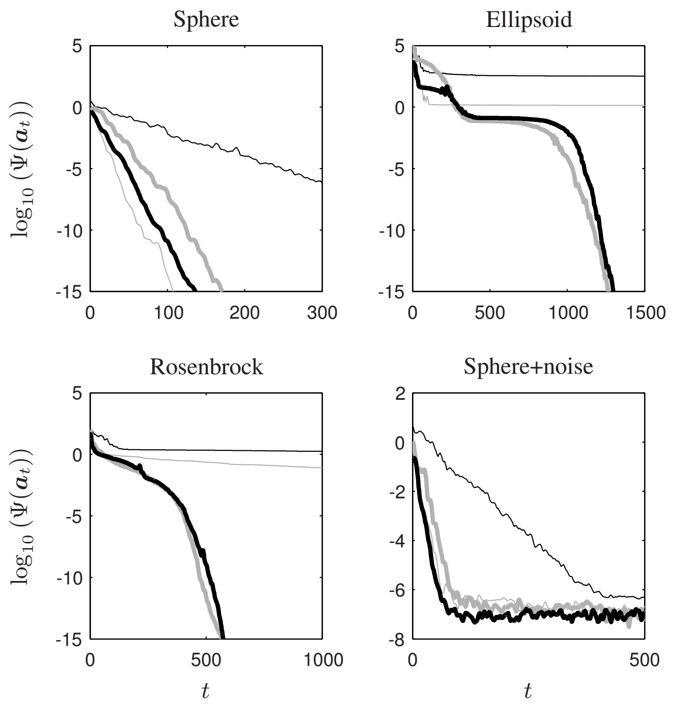

Fig. 2. Objective function value versus generation index for different objective functions. The different lines represent: (1,10)-ES (thin lines), CMA-ES (thick lines), without exponential average (gray lines), with exponential average (black lines).

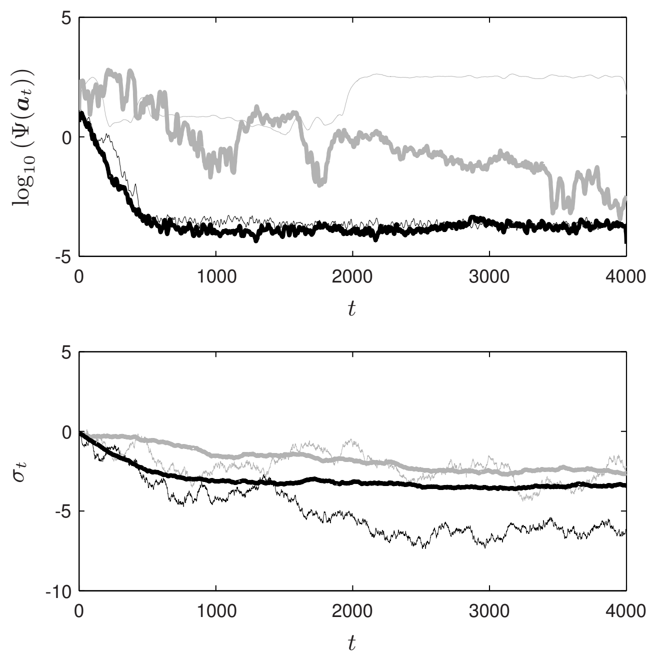

Fig. 3. Objective function value and step size σ versus generation index for the model of objective function encountered in the considered application. The different lines represent: (1,10)-ES (thin lines), CMA-ES (thick lines), without exponential average (gray lines), with exponential average (black lines).

TABLE II
PARAMETERS USED IN SIMULATIONS.

Constant	Value	Purpose
$\mathcal{U}$	4	Number of unknowns
σ_{M}	0.001	Estimation noise
σ_{C}	100	Convergence correlated noise
σ_{T}	0.001	Trajectory dynamics noise
β_{T}	0.1	Trajectory dynamics exponential average
	10^{-6}	Noise level added to the objective function in the noisy Sphere simulation

TABLE III
CONVERGENCE PERCENTAGE (CP) AND MEAN NUMBER OF ITERATIONS (MNI) TO CONVERGENCE FOR DIFFERENT ALGORITHMS AND OBJECTIVE FUNCTIONS.

	(1,10)-ES		(1,10)-ES Exp. Avg.		CMA-ES		CMA-ES Exp. Avg.	
Function	CP	MNI	CP	MNI	CP	MNI	CP	MNI
Sphere	100	96	100	668	100	128	100	103
Ellipsoid	0	Inf	0	Inf	100	1202	100	1258
Rosenbrock	0	Inf	0	Inf	93	515	100	555
Sph. + noise	100	52	100	365	100	66	100	54
App. model	61	4767	94	1066	55	4657	100	420

F. Optimization on noisy objective functions

This section presents a number of numerical simulations to highlight how the exponential average proposed in Section II-D affects the convergence properties of ES algorithms. The CMA-ES is compared to a standard (1,10)-ES algorithm with adaptive step-size update. Both algorithms are executed with and without the exponential average extension. The objective functions used to evaluate the convergence rate in the simulations are the standard functions Sphere, Ellipsoid and Rosenbrock [10]. The algorithms are also simulated on two noisy objective functions, namely noisy Sphere and the model of the objective function encountered in this application, proposed in (18) in Section II-C. The noise-free objective function $\Pi(\cdot)$ is defined as the Sphere, since this function is similar to the objective functions encountered in the considered application. The parameters used in the evaluations are summarized in Table II.

The result from the simulations are presented graphically in Figs. 2 and 3 and numerically in Table III. The graphical presentation shows a typical example of how the objective function value drops as a function of the iteration index t. Fig. 3 also shows the global step size parameter σ as a function of the generation index t. Table III is generated by executing the algorithms on the objective functions 100 times, and recording if they converged as well as the total number of generations required for convergence. The table displays the convergence percentage and the average number of iterations required to converge. The following conclusions can be drawn from this set of results:

- The introduction of the exponential average sometimes reduces the convergence rate for the (1,10)-ES on simple problems.
- For the model of objective function encountered in the proposed application, the exponential average improves the convergence rate by a factor of 4 for the (1,10)-ES and a factor of 10 for CMA-ES. It also increases the convergence probability by 30% for (1,10)-ES and 50% for CMA-ES.
- The (1,10)-ES does not converge for difficult problems, e.g., Rosenbrock and Ellipsoid.
- The exponential average improves the convergence probability on some difficult problems, i.e. Rosenbrock and the application model.
- The exponential average smoothes the step-size evolution for the CMA-ES.

These results indicate that the CMA-ES with the exponential average extension is the best suited optimization method for the considered application due to its robustness to noise on the objective function and its high convergence rate for noisy objective functions.

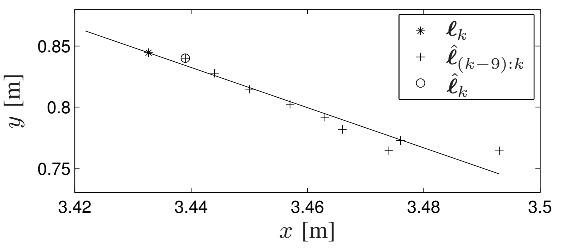

Fig. 4. Extrapolation of the ground truth.

III. Methods for obtaining ground-truth data

The developments presented so far assume knowledge of the actual ground truth (AGT), and will be referred to as the AGT method. As mentioned in the introduction however, it is sometimes impossible to obtain the ground truth for the target position. One option would be to use the position estimate $\hat{\boldsymbol{\ell}}_k$ obtained from the PF as an estimate of the ground truth to form the objective function. This is possible due to the fact that the position estimate obtained form the overall PF is more accurate than that of each individual particle subgroup, as $\hat{\boldsymbol{\ell}}_k$ is calculated using *all* particles. However, a drawback with this method is that the estimated position often lags behind the actual target position. This effect is partly caused by the fact that the particle propagation is influenced by the resampling step, and partly because the parameters for the dynamics model are not optimal during optimization. Experiments showed that these problems lead to a sub-optimal tracker that ultimately becomes unable to keep up with the target, and therefore has a high TLP. Another problem is the high uncertainty in the position estimate during early convergence, as discussed in Section II-C. These two problems can however be solved by using the past few position estimates from the PF in order to extrapolate the current speed and heading of the target. The extrapolated information is subsequently used to form a more robust estimate of the true target position. This principle is illustrated in Fig. 4.

The heading is calculated using a major axis least-squares line fit [31], and the speed is determined from the average distance between the position estimates. The estimated true target position is subsequently calculated by starting at the current position estimate, marked by the circle in Fig. 4, and adding 80% extra travel distance. The resulting point is then projected onto the line, marked by the star in the figure, and is used as the estimated ground truth (EGT). This method will be referred to as the EGT method.

Experiments showed that the EGT method is robust if the type of dynamics model corresponds well to the target dynamics. The PF and ES algorithms will then interact in a mutually beneficial manner, whereby an improvement of the tracking accuracy leads to an improved CMA-ES convergence, and vice versa. However, if a model with too high a model order is used, for example, the PF and the ES will interact in a negative manner and PFES will eventually diverge.

IV. Dynamics models

The dynamics model is required for the transition step in the particle filter. It models how the target is likely to move from one time step to the next. Thus, a model that closely matches the potential movements of the target results in a more robust tracker that has lower RMSE and is less likely to lose track of the target. In the considered example application of ASLT, the target only moves in two dimensions (2D), and the discussion is therefore limited to 2D models only. Note however that the PFES algorithm is not limited to 2D models, but can be used to tune models with higher degrees of freedom.

Dynamics models for 2D tracking can be divided into two main groups: coordinate-uncoupled (CU) and curvilinear (CL). For coordinate-uncoupled models, it is assumed that propagation along the two axes is independent, and for curvilinear models that it is coupled. Both groups can have a different model order, where the noise $\boldsymbol{v}_k$ is added to the position, velocity, or acceleration. A further extension is to band-limit the noise vector to simulate inertia. Models where the noise is band-limited are referred to as time-correlated (TC), and those where it is not as random walk (RW). For coordinate-uncoupled models, this results in one parameter to control noise strength plus one optional parameter to band-limit the noise. For curvilinear models, the number of parameters must be doubled, with one set of parameters required for forward motion and another for turning. By proper adjustment of the time correlation parameter, it is possible to achieve a random walk behavior using a TC model. We have therefore excluded CL-RW models from the models considered here. The resulting possible model type permutations lead to the length $\mathcal{U}$ of the parameter vector $\boldsymbol{a}$ being $\mathcal{U} \in \{1, 2, 4\}$.

As a result from the discretization of the continuous-time model during the derivation of a dynamics model, the elements of $\boldsymbol{a} = [a_1, a_2, \dots, a_{\mathcal{U}}]$ are often transformed and combined before being used in the transition equation. For notational convenience, the transformed values are denoted by the vector $\boldsymbol{b} = [b_1, b_2, \dots, b_{\mathcal{B}}]$, and the equation used in the transformation is defined as $\boldsymbol{b} = \boldsymbol{\Theta}(\boldsymbol{a})$, with $\boldsymbol{\Theta} : \mathbb{R}^{\mathcal{U}} \to \mathbb{R}^{\mathcal{B}}$. The transition and transformation equations for the models considered in this work are listed in Tables IV and V. The vector $\boldsymbol{u}_k$ in the tables is a $(\mathbf{0}, \mathbf{I})$-normally distributed random vector of length $\mathcal{R} = 2$. An in-depth survey of how to derive dynamics models can be found in [19], and it is hence not repeated here.

By analyzing how the time correlation parameters, i.e., a_2 and a_4, affect the transition equations in the two tables, it can be seen that a proper parameter adjustment can change not only between TC and RW for the same model order, but also the model order itself. The transition between model types occurs when a_2 and a_4 tend towards infinity or zero. These relations are summarized in Table VI.

V. Evaluation

A. Simulation setup

An ASLT scenario has been used to evaluate the PFES algorithm. The target is a person speaking and walking inside a medium sized $[5 \times 4 \times 2.3]$m room with a reverberation time of 300ms. The sound inside the room is picked up by an array of 10 microphones located along the walls of the room. The speaker is reciting the text of a book, and an isotropic noise field forms the background noise in the room, resulting in 20dB average SNR at the microphones. The scenario is depicted in Fig. 5. The microphone signals are generated using the image-source method [32], [33], and the target trajectory is randomly generated using concatenated sections of typical human motion trajectories in indoor environments. Two independent signal sequences were generated where the first is an 80 minute sequence used for optimizing the parameter vector, and the second is a 2 minute sequence used for evaluating the performance of the resulting (optimized) tracker. During the evaluation, the RMSE and TLP are calculated and averaged over 10 runs of the short signal sequence.

The PFES algorithm is based on the PF implementation given in Alg. 1, where the likelihood function $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ is computed as a mixture PDF based on the output of a delay-and-sum beamformer as well as the output of a voice activity detector. The algorithm uses a block length of 512 time samples for every position estimate. If

TABLE IV
DEFINITION FOR COORDINATE-UNCOUPLED KINEMATIC MODELS. THE KEY FOR THE TYPE NAMES IS: RANDOM WALK (RW), TIME CORRELATED (TC), POSITION (PO), VELOCITY (VE) AND ACCELERATION (AC).

Type	$\boldsymbol{x}$	Transition equation	$\boldsymbol{b} = \boldsymbol{\Theta}(\boldsymbol{a})$
RWPO	$\begin{bmatrix} x \\ y \end{bmatrix}$	$\boldsymbol{x}_k = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \boldsymbol{x}_{k-1} + b_1 \boldsymbol{u}_k$	$\boldsymbol{b} = \begin{bmatrix} a_1 \end{bmatrix}$
RWVE	$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$	$\boldsymbol{x}_k = \begin{bmatrix} 1 & & T & \\ & 1 & & T \\ & & 1 & \\ & & & 1 \end{bmatrix} \boldsymbol{x}_{k-1} + b_1 \begin{bmatrix} \frac{T}{2} & \\ & \frac{T}{2} \\ 1 & \\ & 1 \end{bmatrix} \boldsymbol{u}_k$	$\boldsymbol{b} = \begin{bmatrix} a_1 \end{bmatrix}$
RWAC	$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix}$	$\boldsymbol{x}_k = \begin{bmatrix} 1 & & T & & \frac{T}{2}^2 & \\ & 1 & & T & & \frac{T}{2}^2 \\ & & 1 & & T & \\ & & & 1 & & T \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \boldsymbol{x}_{k-1} + b_1 \begin{bmatrix} \frac{T}{2}^2 & \\ & \frac{T}{2}^2 \\ T & \\ & T \\ 1 & \\ & 1 \end{bmatrix} \boldsymbol{u}_k$	$\boldsymbol{b} = \begin{bmatrix} a_1 \end{bmatrix}$
TCVE	$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$	$\boldsymbol{x}_k = \begin{bmatrix} 1 & & b_3 & \\ & 1 & & b_3 \\ & & b_2 & \\ & & & b_2 \end{bmatrix} \boldsymbol{x}_{k-1} + b_1 \begin{bmatrix} \frac{T}{2} & \\ & \frac{T}{2} \\ 1 & \\ & 1 \end{bmatrix} \boldsymbol{u}_k$	$\boldsymbol{b} = \begin{bmatrix} a_1 \sqrt{1 - \mathrm{e}^{-2a_2 T}} \\ \mathrm{e}^{-a_2 T} \\ \frac{1-\mathrm{e}^{-a_2 T}}{a_2} \end{bmatrix}$
TCAC	$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix}$	$\boldsymbol{x}_k = \begin{bmatrix} 1 & & T & & b_4 & \\ & 1 & & T & & b_4 \\ & & 1 & & b_3 & \\ & & & 1 & & b_3 \\ & & & & b_2 & \\ & & & & & b_2 \end{bmatrix} \boldsymbol{x}_{k-1} + b_1 \begin{bmatrix} \frac{T}{2}^2 & \\ & \frac{T}{2}^2 \\ T & \\ & T \\ 1 & \\ & 1 \end{bmatrix} \boldsymbol{u}_k$	$\boldsymbol{b} = \begin{bmatrix} a_1 \sqrt{1 - \mathrm{e}^{-2a_2 T}} \\ \mathrm{e}^{-a_2 T} \\ \frac{1-\mathrm{e}^{-a_2 T}}{a_2} \\ \frac{a_2 T - 1 + \mathrm{e}^{-a_2 T}}{a_2^2} \end{bmatrix}$

TABLE V
DEFINITION FOR CURVILINEAR KINEMATIC MODELS. THE KEY FOR THE TYPE NAMES IS: RANDOM WALK (RW), TIME CORRELATED (TC), VELOCITY (VE), ACCELERATION, (AC), TURN (TU) AND TURN RATE (RA). THE PARAMETERS v AND θ REPRESENT THE ABSOLUTE MOTION VELOCITY AND TURNING ANGLE, RESPECTIVELY.

Type	$\boldsymbol{x}$	Transition equation	$\boldsymbol{b} = \boldsymbol{\Theta}(\boldsymbol{a})$
TCVE TCTU	$\begin{bmatrix} x \\ y \\ v \\ \theta \end{bmatrix}$	$\begin{bmatrix} v_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} b_2 & \\ & b_4 \end{bmatrix} \begin{bmatrix} v_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} b_1 & \\ & b_3 \end{bmatrix} \boldsymbol{u}_k$ $\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + T v_k \begin{bmatrix} \cos \theta_k \\ \sin \theta_k \end{bmatrix}$	$\boldsymbol{b} = \begin{bmatrix} a_1 \sqrt{1 - \mathrm{e}^{-2a_2 T}} \\ \mathrm{e}^{-a_2 T} \\ a_3 \sqrt{1 - \mathrm{e}^{-2a_4 T}} \\ \mathrm{e}^{-a_4 T} \end{bmatrix}$
TCAC TCTU	$\begin{bmatrix} x \\ y \\ v \\ \dot{v} \\ \theta \end{bmatrix}$	$\begin{bmatrix} v_k \\ \dot{v}_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} 1 & b_3 & \\ & b_2 & \\ & & b_5 \end{bmatrix} \begin{bmatrix} v_{k-1} \\ \dot{v}_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} Tb_1 & \\ b_1 & \\ & b_4 \end{bmatrix} \boldsymbol{u}_k$ $\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + T v_k \begin{bmatrix} \cos \theta_k \\ \sin \theta_k \end{bmatrix}$	$\boldsymbol{b} = \begin{bmatrix} a_1 \sqrt{1 - \mathrm{e}^{-2a_2 T}} \\ \mathrm{e}^{-a_2 T} \\ \frac{1-\mathrm{e}^{-a_2 T}}{a_2} \\ a_3 \sqrt{1 - \mathrm{e}^{-2a_4 T}} \\ \mathrm{e}^{-a_4 T} \end{bmatrix}$
TCVE TCRA	$\begin{bmatrix} x \\ y \\ v \\ \theta \\ \dot{\theta} \end{bmatrix}$	$\begin{bmatrix} v_k \\ \theta_k \\ \dot{\theta}_k \end{bmatrix} = \begin{bmatrix} b_2 & & \\ & 1 & b_5 \\ & & b_4 \end{bmatrix} \begin{bmatrix} v_{k-1} \\ \theta_{k-1} \\ \dot{\theta}_{k-1} \end{bmatrix} + \begin{bmatrix} b_1 & \\ & Tb_3 \\ & b_3 \end{bmatrix} \boldsymbol{u}_k$ $\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + T v_k \begin{bmatrix} \cos \theta_k \\ \sin \theta_k \end{bmatrix}$	$\boldsymbol{b} = \begin{bmatrix} a_1 \sqrt{1 - \mathrm{e}^{-2a_2 T}} \\ \mathrm{e}^{-a_2 T} \\ a_3 \sqrt{1 - \mathrm{e}^{-2a_4 T}} \\ \mathrm{e}^{-a_4 T} \\ \frac{1-\mathrm{e}^{-a_4 T}}{a_4} \end{bmatrix}$
TCAC TCRA	$\begin{bmatrix} x \\ y \\ v \\ \dot{v} \\ \theta \\ \dot{\theta} \end{bmatrix}$	$\begin{bmatrix} v_k \\ \dot{v}_k \\ \theta_k \\ \dot{\theta}_k \end{bmatrix} = \begin{bmatrix} 1 & b_3 & & \\ & b_2 & & \\ & & 1 & b_6 \\ & & & b_5 \end{bmatrix} \begin{bmatrix} v_{k-1} \\ \dot{v}_{k-1} \\ \theta_{k-1} \\ \dot{\theta}_{k-1} \end{bmatrix} + \begin{bmatrix} Tb_1 & \\ b_1 & \\ & Tb_4 \\ & b_4 \end{bmatrix} \boldsymbol{u}_k$ $\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + T v_k \begin{bmatrix} \cos \theta_k \\ \sin \theta_k \end{bmatrix}$	$\boldsymbol{b} = \begin{bmatrix} a_1 \sqrt{1 - \mathrm{e}^{-2a_2 T}} \\ \mathrm{e}^{-a_2 T} \\ \frac{1-\mathrm{e}^{-a_2 T}}{a_2} \\ a_3 \sqrt{1 - \mathrm{e}^{-2a_4 T}} \\ \mathrm{e}^{-a_4 T} \\ \frac{1-\mathrm{e}^{-a_4 T}}{a_4} \end{bmatrix}$

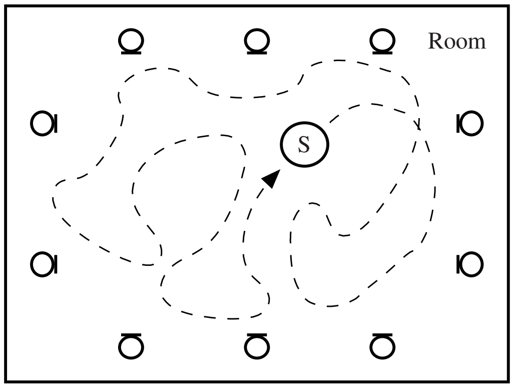

Fig. 5. Scenario used in the evaluation. The speaker S moves in the room along a closed trajectory (example trajectory shown here). Ten microphones are placed along the walls to pick up the sound emitted by the speaker.

TABLE VI
MODEL CHANGES AS A FUNCTION OF TIME CORRELATION PARAMETERS.

Model	$a_2 \to \infty$	$a_2 \to 0$
TCVE	RWPO	RWVE
TCAC	RWVE	RWAC
TCVE-TCTU	RWPO-TCTU	RWVE-TCTU
TCAC-TCTU	RWVE-TCTU	RWAC-TCTU
TCVE-TCRA	RWPO-TCRA	RWVE-TCRA
TCAC-TCRA	RWVE-TCRA	RWAC-TCRA
Model	$a_4 \to \infty$	$a_4 \to 0$
TCVE-TCTU		TCVE-RWTU
TCAC-TCTU		TCAC-RWTU
TCVE-TCRA	TCVE-RWTU	TCVE-RWRA
TCAC-TCRA	TCAC-RWTU	TCAC-RWRA

necessary, readers are referred to [4] for more information on this specific PF implementation.

In the ASLT literature, the most common model is the CU-TCVE, and other similar models such as the Langevin model [4], [34], since these fit quite well with human motion dynamics. On the basis of standard formulae from the laws of kinematic physics, it can also be shown that the CL-TCVE-TCRA model represents a particularly good fit among the curvilinear models considered here. Furthermore, according to [35, p. 202], the use of an acceleration variable in the considered model is only of value when a velocity measurement is available, which is not the case with the considered tracking scenario. Again, this would suggest that the best results should be achieved by the coordinate-uncoupled models CU-RWVE and CU-TCVE, and by the curvilinear model CL-TCVE-TCRA.

B. Implementation

The overall computational burden of the CMA-ES algorithm is low compared to the PF, mainly due to the low update rate. Furthermore, the computational complexity in each iteration of the CMA-ES is kept low through the use of the fast sorting algorithm heap-sort [36], and by using the Mersenne Twister algorithm [37] and the Ziggurat method [38] for random number generation. The PF part of the implementation is based on the one presented in [39].

The PFES algorithm was implemented in C and executed on a standard PC. The numerical parameters used for the simulation settings are summarized in Table VII. Measurements showed an average computational load of around 500M floating point operation per second. This corresponds to around 20% CPU utilization on a modern 2.4GHz PC if the algorithm is running in real-time.

TABLE VII
NUMERICAL PARAMETERS USED IN THE SIMULATIONS.

Parameter	Description	Value		
N	Number of particles	500		
β	Forgetting factor for calculating $\overline{\boldsymbol{a}}_t$	0.2		
G	Position estimates per generation	70		
K	Samples to calculate RMSE and TLP	3125		
ε_{U}	Threshold for lost track	0.5m		
ε_{L}	Threshold for resumed tracking	0.3m		
	Sample frequency for audio signals	16kHz		
	Block length for PF algorithm	512		
	Position estimates for EGT method	9		
	Overshoot for EGT method	80%		
$\mathcal{U}$	Number of parameters	1	2	4
M	Number of parents	4	6	8
L	Number of children	8	12	16
P	Particles per group	62	42	31

C. Dynamic tracking

To gain insight into the dynamical properties of the different models listed in Section IV, the PF algorithm was executed using the optimized parameters on the evaluation data sequence. A small section of the trajectory is displayed in Fig. 6, along with a plot of the microphone signal at one of the microphones. The figure shows the tracking in the x-dimension as a function of time during two speech gaps, for each considered model. During the first gap, the target makes a maneuver, and during the second, the target continues to propagate in the same direction throughout the duration of the gap.

There are two different properties of a good tracker: first, it should minimize the deviations from the trajectory during a gap where the target does not make a maneuver, and second, it should be able to re-localize the target quickly if a maneuver is made during an observation gap. The standard deviation of the particle set during a gap reveals if an optimized tracker performs well. Spreading the particles when no observations are available is what allows the PF to successfully resume tracking in cases where the target makes a maneuver during the gap. Hence, if the model fits the target dynamics properly, the tracker can afford a smaller standard deviation during a gap without risking to loose the target. This can also be seen in how much the estimated target position deviates from the trajectory during tracking, and how quickly it jumps back to the trajectory after a gap. Thus, looking at the figure, it can be seen that the good performers are CU-RWVE, CU-TCVE, CU-TCAC, CL-TCVE-TCRA and CL-TCAC-TCRA.

By examining the target position estimates produced by the CU-RWPO model during a gap, it can be seen that the position estimate starts to deviate during both gaps, which creates a short flat section in the tracking results. This is due to the algorithm effectively stopping to track the target during an observation gap. Furthermore, it has been observed that a model with sub-optimum parameters displays the same problem [4], [40]. On the other hand, none of the other models displays this behavior. The conclusion drawn from this is that models that have the ability to track velocity can efficiently bridge observation gaps, provided that the model parameters are suitably optimized.

D. Convergence to optimum

The convergence of the PFES can be experimentally tested by calculating the RMSE and TLP for different combinations of the model parameter values, and then generate plots of the error surfaces. These plots can be used to determine if PFES effectively converges to an optimum solution. Due to the computational complexity of this approach, it is however only feasible to do this for models with one

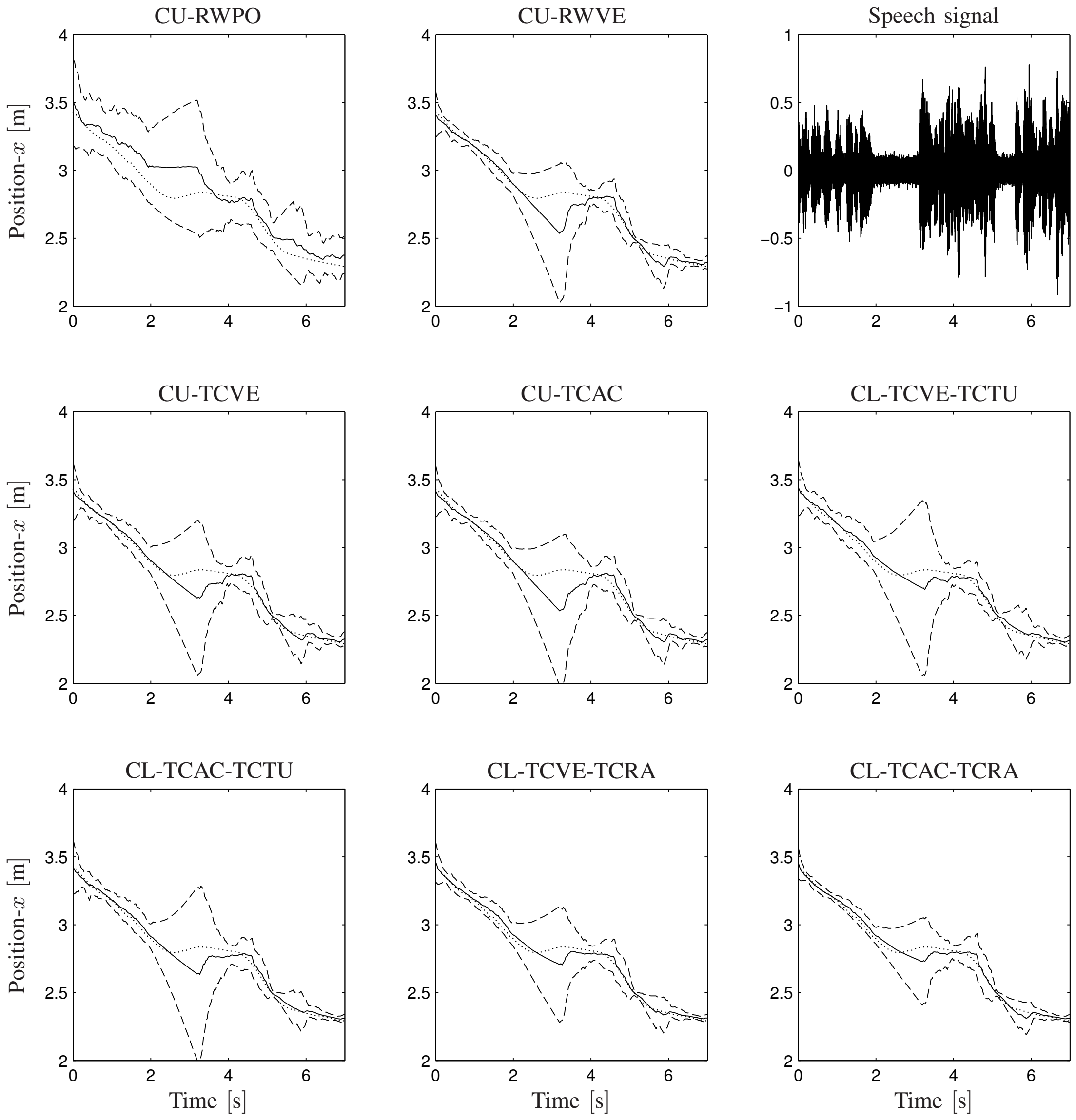

Fig. 6. Tracking results as a function of time with optimized model parameters. The solid and dotted lines are the true source position $\boldsymbol{\ell}_k$ and estimated source position $\hat{\boldsymbol{\ell}}_k$, respectively, and the dashed lines represent the standard deviation ς_k of the particle set. The speech signal at one of the microphones is plotted in the upper right corner.

or two parameters. Examples of plots for models with one and two parameters are depicted in Figs. 7 and 8. The convergence path for the PFES algorithm is included for the surface plots.

These figures show that the algorithm converges to a minimum for both the TLP and RMSE parameters, although it should be noted that the solution returned by the PFES algorithm corresponds to a trade-off between the RMSE and TLP parameters for other types of model representations where both surfaces do not happen to share the same minimum. The first value of the convergence path cannot be graphically represented since the algorithm is randomly initiated over the range of meaningful values of the objective function space. Thus, the first point shown on the path corresponds to the first value returned by CMA-ES, i.e., $\overline{\boldsymbol{a}}_1$, and is marked by a circle in the surface plots.

If the model parameters are extremely poorly tuned, the PF will completely fail to track the target. However, due to the limited region over which the tracking takes place, the TLP measurement will not reach 100% since the target will occasionally pass close to the estimated source position. The limit above which the TLP is considered unreliable is around 80% and is represented by the shaded regions in the surface plots.

In the surface plots, the data used to generate the surfaces is calculated using the PFES, with the CMA-ES part disabled. The simulation time to generate data for a surface plot is around one week. In contrast, the PFES with CMA-ES enabled finds the minimum of the surface in twelve to fifteen minutes of simulation time.

While it is possible to draw some conclusions regarding the convergence properties of the PFES algorithm by analyzing its convergence path with respect to the RMSE and TLP surface plots, it of course does not constitute a formal proof of convergence. The plots do however confirm that PFES is able to efficiently detect the optimum of the corresponding surfaces in the considered tracking scenario. The convergence properties of evolutionary strategies still represent an active field of research, and whilst many advancements have been made in this area [41]–[43], a formal convergence proof for the CMA-ES is yet to be derived [44]. However, the numerical evaluations presented here indicate that the CMA-ES performs well in the considered application.

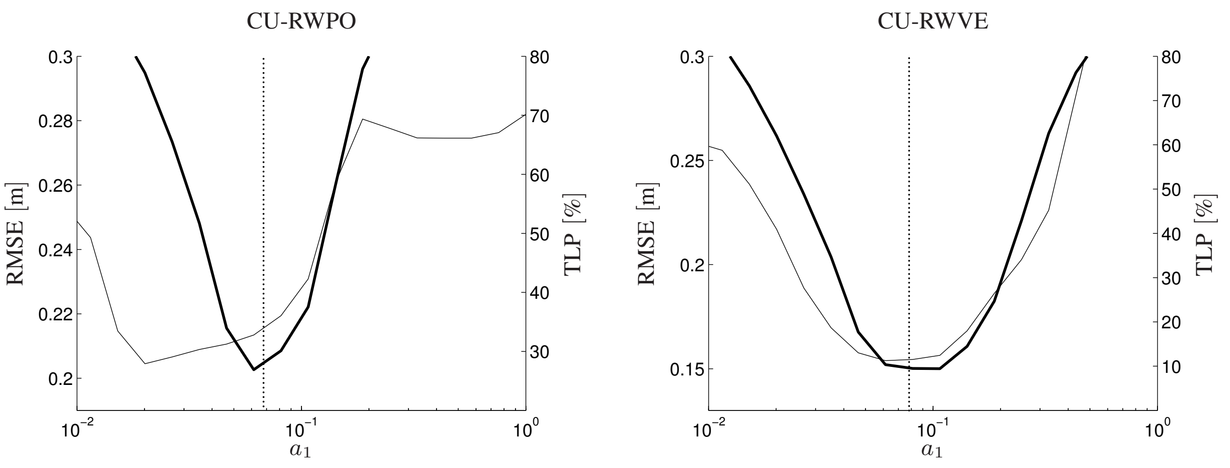

Fig. 7. TLP (thick line) and RMSE (thin line) as a function of the parameter a_1 for RW models. The dotted vertical line indicates the final optimum value from the PFES optimization algorithm.

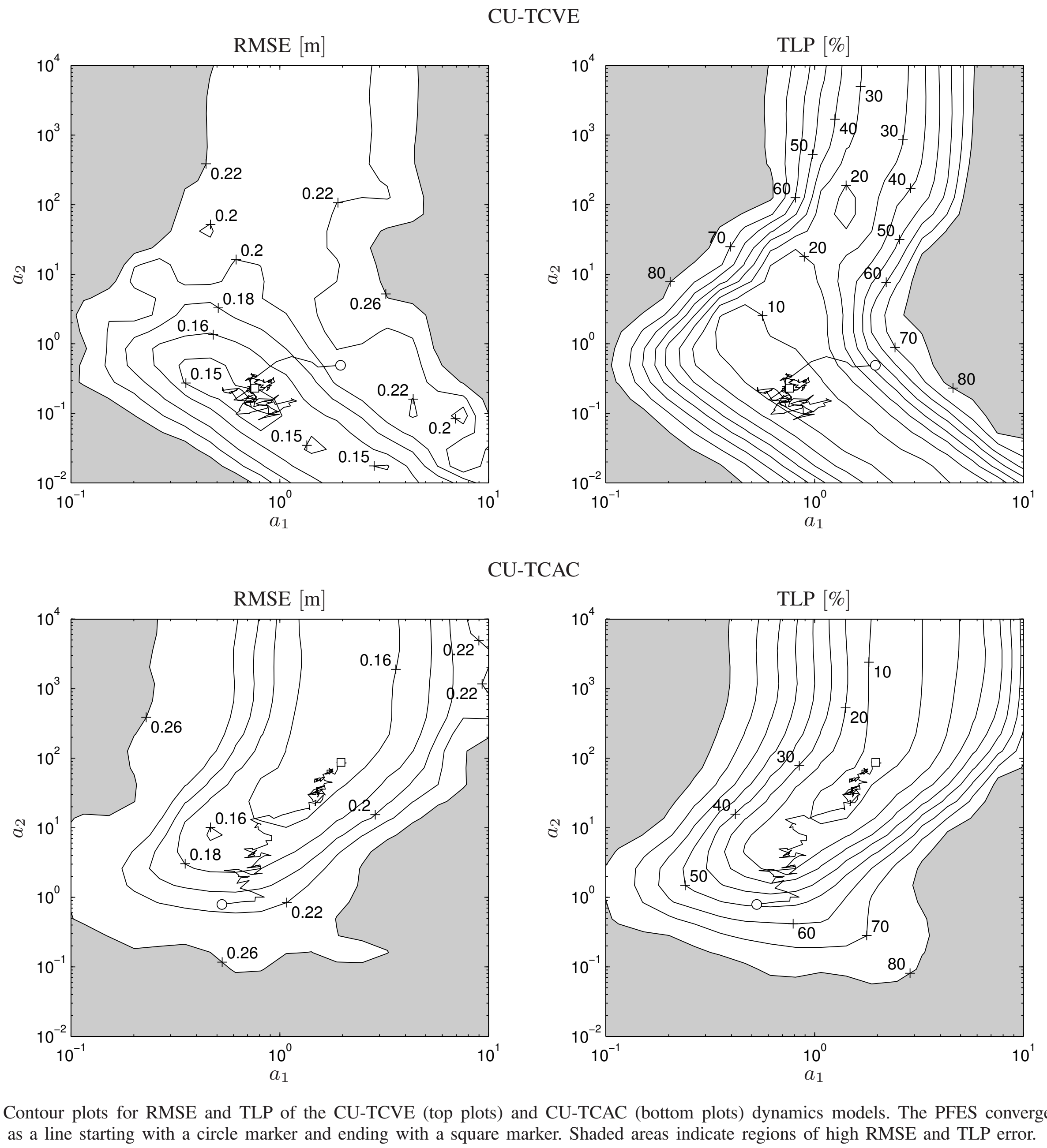

Fig. 8. Contour plots for RMSE and TLP of the CU-TCVE (top plots) and CU-TCAC (bottom plots) dynamics models. The PFES convergence path is displayed as a line starting with a circle marker and ending with a square marker. Shaded areas indicate regions of high RMSE and TLP error.

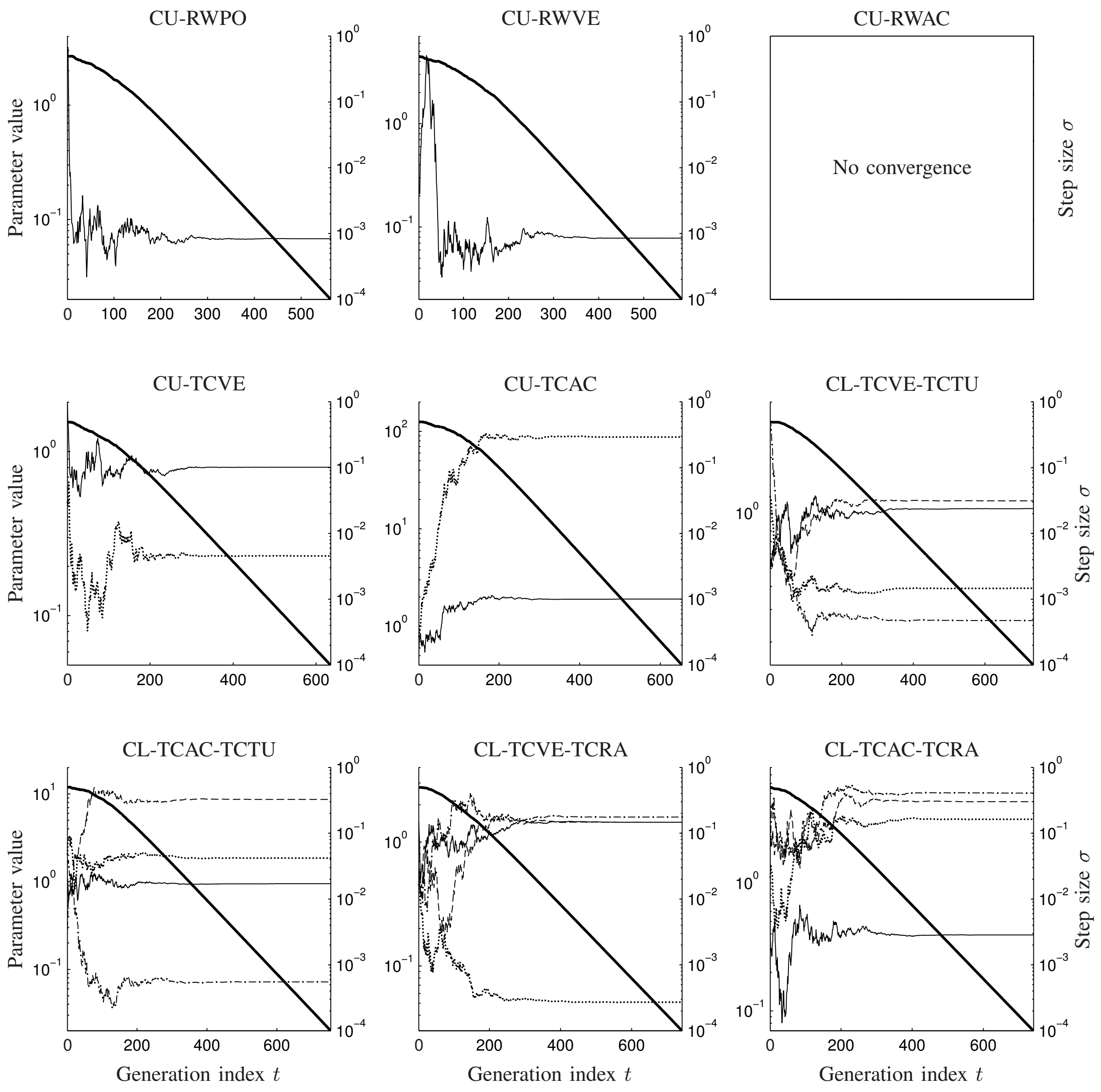

Fig. 9. Step size σ and value of parameters in $\boldsymbol{a}$ as a function of the generation index t, during the optimization of all models. The thick line corresponds to the step size and the thin lines to the parameters in $\boldsymbol{a}$, including the velocity noise a_1 (solid line), velocity correlation a_2 (dotted line), turn noise a_3 (dashed line), and turn correlation a_4 (dash-dotted line).

E. Adaptation of the CMA-ES

Adaptation curves for the CMA-ES were generated for the models listed in Section IV using the AGT method for calculating the objective function. The curves are displayed in Fig. 9. The figure indicates a number of interesting properties of the dynamics models and the algorithm:

- The number of iterations needed to reach convergence is proportional to the number of model parameters $\mathcal{U}$, i.e., the length of the vector $\boldsymbol{a}$, and does not appear to be related to the model type. Hence, it can be assumed that in this application, the major influence on the convergence rate is the number of unknowns and not the shape of the objective function.
- Models with too high a model order tend towards random-walk models with lower model order, i.e., the TC parameter becomes high. This can be clearly seen in the CU-TCAC model, which converges to CU-RWVE. This is further confirmed by how the model responds to gaps as seen in Fig. 6, and by the shape of the TLP and RMSE surface plots in Fig. 8. This tendency is also visible in the TCAC parameter a_2 for the CL-TCAC-TCRA model, and in the TCRA parameter a_4 for the CL-TCAC-TCRA model.
- The CU-RWAC model fails to converge. The reason is that it is impossible to track acceleration using a CU model when no velocity observation is present. This is again confirmed by the TLP surface plot in Fig. 8, which shows that small values of a_2 result in tracking failure, and by the fact that the CL-TCAC model converges to CL-RWVE.
- The curvilinear models CL-TCAC-TCTU and CL-TCVE-TCTU have a very low parameter a_4, which indicates that the model order used to represent the turn variable may be too low. This is also highlighted by the high standard deviation during observation gaps obtained for these models, as seen in Fig. 6.

- All curvilinear models have a high noise level for the turn parameter a_3. This results from the high maneuverability of humans. In general [19], curvilinear models will provide a better fit for targets with less maneuverability, such as vehicles and aircraft.

F. Numerical performance results

The optimum parameters for all models, using both the AGT and the EGT methods for the objective function, have been calculated using PFES, and the performance of the resulting PF algorithms has been evaluated with regards to TLP and RMSE. The results from this evaluation are tabulated in Table VIII and confirm the hypothesis stated earlier that the best results should be achieved by the coordinate-uncoupled models CU-RWVE and CU-TCVE, and by the curvilinear model CL-TCVE-TCRA. The table further shows that these models achieve a proper convergence when the estimated source position is used in the computation of the objective function (EGT method), and that models that do not fit the dynamics well sometimes result in convergence failure. These results again confirm that TC models with an unnecessarily high model order, i.e., CU-TCAC and CL-TCAC-TCRA, converge to lower-order RW models.

For comparison, Table VIII also includes results for a PF used in conjunction with (1,10)-ES with exponential average, using the AGT method to calculate the objective function. The results show that this method can be used to identify optimum parameters for dynamics models with a low parameter count that fit the target dynamics well, i.e., CU-TCVE, but in general leads to a higher track loss factor, as demonstrated in Table VIII. Furthermore, experiments showed that the method is not stable enough to be used in conjunction with the EGT method for calculating the objective function.

TABLE VIII
PARAMETER VALUES, TLP AND RMSE FOR OPTIMIZED MODELS USING BOTH THE AGT AND THE EGT METHOD TO CALCULATE THE OBJECTIVE FUNCTION VALUE. FOR COMPARISON, THIS TABLE ALSO INCLUDES THE RESULTS FROM (1,10)-ES WITH EXPONENTIAL AVERAGE.

Results for PFES using the AGT method			
Model	$\overline{\rho}$ [%]	$\overline{\varepsilon}$ [m]	$\boldsymbol{a}_{\text{OPT}}$
CU-RWPO	25.48	0.22	$[0.068]$
CU-RWVE	7.89	0.15	$[0.078]$
CU-RWAC	Convergence not possible		
CU-TCVE	6.98	0.15	$[0.80 \quad 0.23]^{\mathrm{T}}$
CU-TCAC	8.22	0.15	$[1.90 \quad 87.44]^{\mathrm{T}}$
CL-TCVE-TCTU	12.57	0.18	$[1.06 \quad 0.39 \quad 1.16 \quad 0.26]^{\mathrm{T}}$
CL-TCAC-TCTU	15.91	0.17	$[0.95 \quad 1.86 \quad 8.65 \quad 0.07]^{\mathrm{T}}$
CL-TCVE-TCRA	10.85	0.16	$[1.45 \quad 0.05 \quad 1.44 \quad 1.59]^{\mathrm{T}}$
CL-TCAC-TCRA	11.04	0.16	$[0.39 \quad 3.14 \quad 4.32 \quad 5.05]^{\mathrm{T}}$
Results for PFES using the EGT method			
Model	$\overline{\rho}$ [%]	$\overline{\varepsilon}$ [m]	$\boldsymbol{a}_{\text{OPT}}$
CU-RWPO	28.27	0.21	$[0.052]$
CU-RWVE	8.53	0.15	$[0.077]$
CU-RWAC	Convergence not possible		
CU-TCVE	6.82	0.15	$[0.83 \quad 0.21]^{\mathrm{T}}$
CU-TCAC	No convergence, bad model fit		
CL-TCVE-TCTU	No convergence, bad model fit		
CL-TCAC-TCTU	15.94	0.18	$[1.38 \quad 1.85 \quad 1.65 \quad 0.47]^{\mathrm{T}}$
CL-TCVE-TCRA	11.73	0.17	$[0.59 \quad 0.51 \quad 1.91 \quad 3.99]^{\mathrm{T}}$
CL-TCAC-TCRA	13.52	0.16	$[0.49 \quad 3.33 \quad 2.10 \quad 2.08]^{\mathrm{T}}$
Results for PF + (1,10)-ES using the AGT method			
Model	$\overline{\rho}$ [%]	$\overline{\varepsilon}$ [m]	$\boldsymbol{a}_{\text{OPT}}$
CU-RWPO	26.22	0.22	$[0.071]$
CU-RWVE	8.64	0.15	$[0.095]$
CU-RWAC	Convergence not possible		
CU-TCVE	6.45	0.15	$[0.50 \quad 0.34]^{\mathrm{T}}$
CU-TCAC	19.44	0.17	$[0.47 \quad 4.30]^{\mathrm{T}}$
CL-TCVE-TCTU	14.75	0.17	$[0.59 \quad 0.46 \quad 3.56 \quad 0.24]^{\mathrm{T}}$
CL-TCAC-TCTU	24.46	0.16	$[0.49 \quad 2.27 \quad 1.10 \quad 0.61]^{\mathrm{T}}$
CL-TCVE-TCRA	15.80	0.16	$[0.40 \quad 0.23 \quad 1.52 \quad 1.95]^{\mathrm{T}}$
CL-TCAC-TCRA	17.49	0.18	$[0.52 \quad 1.22 \quad 1.11 \quad 1.70]^{\mathrm{T}}$

VI. CONCLUSION

This article describes a new method for the parameter optimization of dynamics models involved in target tracking applications. The proposed algorithm combines an evolutionary strategy using covariance matrix adaptation with a sequential Monte Carlo approach to dynamic target tracking. Together with the efficiency of CMA-ES, the adaptive nature of particle filtering leads to an algorithm which is capable of producing optimum model parameters even if the true source position is not known. Experimental results were obtained from a series of extensive simulations for the particular application of acoustic source tracking, demonstrating how the proposed method can be used to evaluate the suitability of various dynamics models on the basis of their performance using optimum parameters.

A natural extension of the proposed algorithm is to allow it to adapt the dynamics model to the instantaneous maneuvers of the target, in a way similar to how the PF is able to track the target's state. This could potentially yield an increased tracking accuracy and reduced track loss. Preliminary experiments indicate that one promising approach to this purpose is to enforce a minimum limit on the allowed step size parameter. Another approach which has also shown some success is to completely freeze the covariance matrix and step-size updates, and only allow mutation.

ACKNOWLEDGEMENTS

This work was supported by National ICT Australia (NICTA). NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

REFERENCES

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, February 2002.

[2] A. Giremus, J.-Y. Tourneret, and V. Calmettes, "A particle filtering approach for joint detection/estimation of multipath effects on GPS measurements," *IEEE Transactions on Signal Processing*, vol. 55, no. 4, pp. 1275–1285, April 2007.

[3] R. Karlsson and N. Bergman, "Auxiliary particle filters for tracking a maneuvering target," in *IEEE Conference on Decision and Control*, vol. 4, Sydney, NSW, Australia, December 2000, pp. 3891–3895.

[4] E. A. Lehmann and A. M. Johansson, "Particle filter with integrated voice activity detection for acoustic source tracking," *EURASIP Journal on Advances in Signal Processing*, 2007, article ID 50870, 11 pages.

[5] D. W. Pace, M. Mallic, and W. Eldredge, "Spectral feature-aided multi-target multi-sensor passive sonar tracking," in *IEEE/MTS Oceans 2003*, vol. 4, September 2003, pp. 2120–2126.

[6] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings F, Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, 1993.

[7] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies—A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[8] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing*, vol. 1, no. 2-3, pp. 235–306, 2002.

[9] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[10] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[11] N. Hansen and S. Kern, "Evaluating the CMA evolution strategy on multimodal test functions," in *Parallel Problem Solving from Nature, PPSN VIII*, X. Yao, Ed., vol. 3242, December 2004, pp. 282–291.

[12] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple SVM parameters," in *Proceedings of the European Symposium on Artificial Neural Networks*, vol. 64, Bruges, Belgium, April 2004, pp. 107–117.

[13] S. D. Müller, I. Mezić, J. H. Walther, and P. Koumoutsakos, "Transverse momentum micromixer optimization with evolution strategies," *Computers and Fluids*, vol. 33, no. 4, pp. 521–531, May 2004.

[14] K. C. C. Chan, V. Lee, and H. Leung, "Generating fuzzy rules for target tracking using a steady-state genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 3, pp. 189–200, September 1997.

[15] P. J. Fleming and R. C. Purshouse, "Evolutionary algorithms in control systems engineering: a survey," *Control Engineering Practice*, vol. 10, no. 11, pp. 1223–1241, April 2002.

[16] P. Cerveri, N. Lopomo, A. Pedotti, and G. Ferrigno, "Derivation of centers and axes of rotation for wrist and fingers in a hand kinematic model: methods and reliability results," *Annals of Biomedical Engineering*, vol. 33, no. 3, pp. 402–412, March 2005.

[17] C. Igel, W. Erlhagen, and D. Jancke, "Optimization of neural field models," *Neurocomputing*, vol. 36, no. 1–4, pp. 225–233, 2001.

[18] A. Pellecchia, C. Igel, J. Edelbrunner, and G. Schoner, "Making driver modeling attractive," *IEEE Intelligent Systems*, vol. 20, no. 2, pp. 8–12, 2005.

[19] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking—Part I. Dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, October 2003.

[20] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments—A survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, June 2005.

[21] I. Rechenberg, "Online biography for Prof. Dr.-Ing. Ingo Rechenberg," Web Page, http://www.bionik.tu-berlin.de/institut/n2rechenb.html.

[22] H. P. Schwefel, "Evolutionsstrategie und numerische Optimierung," Ph.D. dissertation, Technical University of Berlin, Germany, 1975.

[23] I. Rechenberg, *Simulationsmethoden in der Medizin und Biologie*. Berlin: Springer Verlag, 1978, ch. Evolutionsstrategien, pp. 83–114.

[24] ——, "Evolutionsstrategie—Optimierung technischer Systeme nach Prinzipien der biologischen Evolution," Ph.D. dissertation, Technical University of Berlin, Germany, 1971, published in 1973 by Fromman-Holzboog.

[25] H.-P. Schwefel, *Numerical optimization of computer models*. John Wiley and Sons, Inc., 1981.

[26] A. Ostermeier, A. Gawelczyk, and N. Hansen, "Step-size adaptation based on non-local use of selection information," in *Parallel Problem Solving from Nature, PPSN IV*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds., 1994, pp. 189–198.

[27] H.-G. Beyer, "Mutate large, but inherit small! On the analysis of rescaled mutations in $(\tilde{1}, \tilde{\lambda})$-ES with noisy fitness data," in *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*. London, UK: Springer-Verlag, 1998, pp. 109–118.

[28] K. Uosaki, Y. Kimura, and T. Hatanaka, "Evolution strategies based particle filters for state and parameter estimation of nonlinear models," *IEEE Congress on Evolutionary Computation*, vol. 1, pp. 884–890, June 2004.

[29] N. M. Kwok, F. Gu, and W. Zhou, "Evolutionary particle filter: re-sampling from the genetic algorithm perspective," in *International Conference on Intelligent Robots and Systems*, August 2005, pp. 2935–2940.

[30] D. V. Arnold and H.-G. Beyer, "Optimum tracking with evolution strategies," *Evolutionary Computation*, vol. 14, no. 3, pp. 291–308, September 2006.

[31] D. I. Warton, I. J. Wright, D. S. Falster, and M. Westoby, "Bivariate line-fitting methods for allometry," *Biological Reviews*, vol. 81, no. 2, pp. 259–291, May 2006.

[32] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, April 1979.

[33] E. A. Lehmann and A. M. Johansson, "Prediction of energy decay in room impulse responses simulated with an image-source model," *Journal of the Acoustical Society of America*, vol. 124, no. 1, pp. 269–277, July 2008.

[34] J. Vermaak and A. Blake, "Nonlinear filtering for speaker tracking in noisy and reverberant environments," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, Salt Lake City, UT, USA, May 2001, pp. 3021–3024.

[35] S. Blackman and R. Popoli, *Design and analysis of modern tracking systems*. Boston: Artech House, 1999.

[36] J. W. J. Williams, "Algorithm 232—Heapsort," *Communications of the ACM*, vol. 7, no. 6, pp. 347–348, 1964.

[37] M. Matsumoto and T. Nishimura, "Mersenne Twister: a 623-dimensionally equidistributed uniform pseudorandom number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, January 1998.

[38] G. Marsaglia and W. W. Tsang, "The Ziggurat method for generating random variables," *Journal of Statistical Software*, vol. 5, no. 8, 2000.

[39] A. Johansson and E. Lehmann, "Real-time implementation of a particle filter with integrated voice activity detector for acoustic speaker tracking," in *IEEE Asia Pacific Conference in Circuits and Systems*, Singapore, December 2006.

[40] E. Lehmann and A. Johansson, "Experimental performance assessment of a particle filter with voice activity data fusion for acoustic speaker tracking," in *Nordic Signal Processing Symposium*, Reykjavik, Iceland, June 2006.

[41] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 96–101, January 1994.

[42] ——, *Convergence Properties of Evolutionary Algorithms*. Kovac, January 1997.

[43] H.-G. Beyer and S. Meyer-Nieberg, "Self-adaptation of evolution strategies under noisy fitness evaluations," *Genetic Programming and Evolvable Machines*, vol. 7, no. 4, pp. 295–328, 2006.

[44] A. Auger, "Convergence results for the (1,λ)-SA-ES using the theory of ϕ-irreducible Markov chains," *Theoretical Computer Science*, vol. 334, pp. 35–69, 2005.