J Glob Optim (2007) 37:381–403
DOI 10.1007/s10898-006-9054-8

ORIGINAL ARTICLE

Mobile Robot Global Localization using an Evolutionary MAP Filter

L. Moreno · S. Garrido · D. Blanco

Received: 30 September 2005 / Accepted: 7 June 2006 /
Published online: 21 July 2006

Abstract A new algorithm based on evolutionary computation concepts is presented in this paper. This algorithm is a non linear evolutive filter known as the Evolutive Localization Filter (ELF) which is able to solve the global localization problem in a robust and efficient way. The proposed algorithm searches stochastically along the state space for the best robot pose estimate. The set of pose solutions (the population) represents the most likely areas according to the perception and motion information up to date. The population evolves by using the log-likelihood of each candidate pose according to the observation and the motion error derived from the comparison between observed and predicted data obtained from the probabilistic perception and motion model. The algorithm has been tested on a mobile robot equipped with a laser range finder to demonstrate the effectiveness, robustness and computational efficiency of the proposed approach.

Keywords Evolutive algorithm · Maximum a posteriory estimate · Non-linear filter · Global localization · Mobile robots

1 Introduction

Localization is a key component in geometrical robot navigation and required to execute successfully a geometrical path generated by a global planner. Two different cases can be distinguished: the re-localization case and the global localization case. *Re-localization problem* tries to keep track of mobile robot pose, where the robot knows its initial position (at least approximately) and therefore has to maintain localized the robot along the given mission. The *global localization problem* does not

L. Moreno · S. Garrido (✉) · D. Blanco
System Engineering and Automation, Carlos III University, calle Universidad, 30, Leganes, Madrid 28911, Spain
e-mail: sgarrido@ing.uc3m.es

assume any knowledge about the robot's initial position and therefore has to globally localize itself. An associated problem is the *robot kidnapping problem*, which tries to determine the mobile robot pose, once the robot has been displaced from its known position to another unknown arbitrary pose without providing any motion estimation information to the mobile robot. In that case, the robot's estimated position is erroneous and therefore has to recover from a serious pose failure to globally localize itself.

The majority of existing algorithms address only the re-localization problem. In this case the small incremental errors produced along the robot motion and the initial knowledge of the robot pose makes classical approaches such as Kalman filters applicable. The Kalman filter for re-localization was introduced in the eighties [4, 6, 16, 19]. If we consider the robot pose estimation as a Bayesian recursive problem, Kalman filters estimate posterior distribution of poses conditioned on sensor data. Based on the Gaussian noise assumption and the Gaussian-distributed initial uncertainty, this method represents posterior distributions by Gaussians. Kalman filters constitute an efficient solution to the re-localization problem. However, the assumptions nature of the uncertainty representation makes Kalman filters not robust in global localization problems.

Three families of algorithms provide a solution to the global localization problem: multi-hypothesis Kalman filters [1, 2, 5, 16, 25], grid-based probabilistic filters [3, 9, 24] and Monte Carlo localization methods [7, 14, 23]. Those methods can be included in a wider scope group of Bayesian estimation methods. This kind of methods are described in detail in Sect. 2 and 3.

This article presents a localization algorithm based on a non-linear filter called the Evolutionary Localization Filter (ELF). ELF solves the global localization robot problem in a robust and efficient way. The algorithm can accommodate arbitrary noise distributions and non-linear state space systems. It operates with raw sensor data, avoiding to extract features from the sensor data. The key idea of ELF is to represent the uncertainty about the robot pose by a set of possible pose estimates weighted by a fitness function derived from the log-likelihood of the pose given the previous estimate. The idea of estimating state recursively using set of solutions is conceptually similar to Monte Carlo localization methods. The difference relies on: the significance of the weight associated to each possible solution included in the set, and in the method the set of solutions evolve in time (to integrate the raw sensor information and the robot motion information). The adaptation engine of the ELF method is based on genetic adaptation ideas applied to the gradient method.

The rest of this article is organized as follows. Section 2 introduces the mathematical derivation of the global localization problem as a Bayesian recursive problem. In Sect. 3, the specific formulations of different existing approaches able to solve this problem, like multi-hypothesis Kalman filter, grid-based probabilistic filters or Monte Carlo filters are revised. Section 4 introduces the basic mechanism of the differential evolutive filter. The fitness function is addressed, and the way of including in this function all available error information is introduced. Posteriorly the ELF algorithm is explained. Section 5 shows the experimental results. This section includes the analysis of the convergence results, the accuracy and robustness, and the computational cost. Finally, a description of related work is done in Sect. 6 and the advantages and disadvantages of the algorithm are discussed in the last section.

2 Bayesian formulation of localization problem

From a Bayesian point of view, the localization problem is basically a probability density estimation problem, where the robot seeks to estimate a posterior distribution over the space of its poses conditioned on the available data. The robot's pose $(x, y, \theta)^T$ at time t will be denoted by x_t, and the data up to time t by Y_t. The posterior probability distribution according to this notation can be written as $p(x_t|Y_t, m)$, where m is the environment model which is known. To alleviate the notation, the term m is not included in the following expressions, $p(x_t|Y_t)$. Sensor data typically comes from two different sources: motion sensors which provide data relating to change of the situation (e.g., odometer readings) and perception sensors which provide data relating to environment (e.g., camera images, laser range scans, ultrasound measures). We refer the former as motions u_i and the latter as observations z_i. Motion $u(t-1)$ refers to the robot displacement in the time interval $[t-1, t]$ as a consequence of the control command given at time $t-1$. We will consider that both types of data arrives alternatively $Y_t = \{z_0, u_0, \ldots, z_{t-1}, u_{t-1}, z_t\}$.

These sensor data can be divided in two groups of data $Y_t \equiv \{Z_t, U_{t-1}\}$ where $Z_t = \{z_0, \ldots, z_t\}$ contains the perception sensor measurements and $U_{t-1} = \{u_0, \ldots, u_{t-1}\}$ contains the odometric information. To estimate the posterior distribution $p(x_t|Y_t)$, probabilistic approaches resort to the *Markov assumption*, which states that future states only depend of the knowledge of the current state and not how the robot got there, i.e., they are independent of past states.

The Bayesian recursive determination of the posterior probability density can be computed in two steps:

- *Measurement update*

  According to the notation and applying the Bayes' rule to the last element of the measurement vector Y_t yields

$$p(x_t|Y_t) = \frac{p(z_t|x_t, Y_{t-1})p(x_t|Y_{t-1})}{p(z_t|Y_{t-1})} = \frac{p(z_t|x_t)p(x_t|Y_{t-1})}{p(z_t|Y_{t-1})} \tag{1}$$

$$p(z_t|Y_{t-1}) = \int_{\Re^n} p(z_t|x_t)p(x_t|Y_{t-1})\mathrm{d}x_t. \tag{2}$$

  since it has been assumed that the observation z_t is conditionally independent of the previous measurements given the state x_t. Expression 1 is referred to as the *measurement update* in the Bayesian recursion. The denominator of 1 is obtained by marginalization in Eq. 2.

- *Prediction*

  The effect of a time step over the state given the observations up to time t is obtained by observing that

$$p(x_{t+1}|Y_t) = \int_{\Re^n} p(x_{t+1}|x_t, u_t, Y_t)p(x_t|Y_t)\mathrm{d}x_t = \int_{\Re^n} p(x_{t+1}|x_t, u_t)p(x_t|Y_t)\mathrm{d}x_t \tag{3}$$

where the assumption that the process x_t is Markovian, and then x_{t+1} is independent of Y_t has been considered. Equations 1, 2 and 3 are the solution to a recursive Bayesian estimation problem. In general, the multidimensional integrals in expressions 2 and 3 have no explicit analytical solutions for nonlinear and non-Gaussian models.

To implement Eqs. 1–3 it is necessary to specify $p(x_{t+1}|u_t, x_t)$ and $p(z_t|x_t)$. The first conditional density $p(x_{t+1}|u_t, x_t)$ is frequently referred to as *probabilistic motion model* and the conditional density $p(z_t|x_t)$ is called *probabilistic sensor model*. These distributions will be obtained from the mobile robot state space model. The state space estimation model is given by

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + v_t \\ z_t &= h(x_t) + e_t \end{aligned} \tag{4}$$

The motion model is a probabilistic generalization of the robot kinematics. The probabilistic motion model describes a posterior density over possible state successors x_{t+1}, given the state x_t and control input u_t . Motion noise is typically modelled by Gaussian noise added to the robot motion components in the odometric measurements. The probabilistic observation model $p(z_t|x_t)$ expresses the uncertainty in the environment information perceived by the mobile robot sensors. Since we assume that the robot is given a map of the environment, and that the robot pose is x_t, z_t can be computed to obtain the expected distance to be observed by the sensor. The probabilistic perception model $p(z_t|x_t)$ describes the posterior density over possible sensor measurement z_t. Noise v_t and e_t is typically modelled by a Gaussian noise added to the expected distance.

According to these probabilistic models, the recursive expressions to be calculated at each iteration of the recursive Bayesian filter 1–3 are given by

$$p(x_t|Y_t) = \alpha_t^{-1} p_e(z_t - h(x_t))p(x_t|Y_{t-1}) \tag{5}$$

$$\alpha_t = \int\limits_{R^n} p_e(z_t - h(x_t))p(x_t|Y_{t-1})\mathrm{d}x_t \tag{6}$$

$$p(x_{t+1}|Y_t) = \int\limits_{R^n} p_v(x_{t+1} - f(x_t, u_t))p(x_t|Y_t)\mathrm{d}x_t \tag{7}$$

where $p_e(\cdot)$ and $p_v(\cdot)$ are Gaussian probability density expressions obtained from the probabilistic observation and motion models. The algorithm 5–7 requires to solve two generalized integrals to obtain the posterior probability distribution.

The posterior is a general solution to the estimation problem but rather difficult to obtain and to manage for general non-linear and non-Gaussian problems. Each candidate parameter value in $\Re^n$ yields a value of $p(x|y)$ reflecting the posterior probability of the robot pose given the data up to time t. The complete posterior density function contains all information available about the system, but to obtain an estimate it is required to specify the parameter value. This posterior needs to be weighted according to a given criterium to determine an estimate $\hat{x}$ of the true parameter value, that is the estimated pose value. Each choice of cost function lead us to different estimators. Two common choices are the mean-square error and the maximum a posteriori estimators.

The minimum mean-square estimate is defined by

$$\hat{x}_{MS} = \arg\min_{x_t^*} \int_{\Re^n} (x_t - x_t^*)^T (x_t - x_t^*) p(x_t|Y_t) \mathrm{d}x$$
$$= \int_{\Re^n} x_t p(x_t|Y_t) \mathrm{d}x \tag{8}$$

Expression 8 yields that the optimal estimate in the mean-square sense is the conditional mean. This estimate is also referred to as the *conditional mean estimate*. Since the posterior probability distribution is multi-modal in the global localization problem this estimate is not convenient. Another common estimator, and much more interesting for global localization, is the *maximum a posteriori*. This estimator can be defined as $\hat{x}^{\mathrm{MAP}} = \arg\max_x p(x_t|Y_t)$. This estimator, chosen to develop the ELF algorithm, is explained in depth in Sect. 4.

3 Existing solutions

Depending on the method of representing the probability density function $p(x_t|Y_t)$ different classes of filters are obtained. Before presenting the Evolutionary filtering method, it is convenient to consider the existing methods of obtaining the *posterior probability density* $p(x_t|Y_t)$ which lead us to substantially different methods with very different properties. Equations 2 and 3 of the recursive Bayesian filter require to evaluate integrals. This integrals can initially be numerically evaluated since the integrand consists of analytically known functions defined in the problem model. But, independently of the numerical integration method used, the prior at the next iteration is only known approximately. Future iterations will have integrands that are not analytically known. The main arguments against that numerical integration in a n dimensional Euclidean space is the computational burden. It can be shown that, for a given level of accuracy, the computational requirements grow exponentially with the state dimension. This exponential relation is usually referred as the *curse of dimensionality*.

There are a number of approximation schemes that will turn the Bayesian functional propagation into a tractable equivalent. The common idea of these schemes is that the propagation of the continuous probability density function $p(x_t|Y_t)$ is replaced by a propagation of the probability in a finite set of simple probability elements spread over the region of interest in the state space. Some different approaches that will end up as a feasible algorithm are:

- *Mixture of Gaussians*

  The whole distribution is approximated as a sum of a set of weighted Gaussian distributions. The probability at a given point is the weighted sum of the probability induced by each Gaussian distribution at that point (see Fig. 1a).

$$p(x_t|Y_t) \approx \sum_{i=1}^{N} \gamma_i \mathcal{N}(x_t; \mu_i, \Sigma_i) \tag{9}$$

  This approach takes advantage of the property of Gaussian distributions that multiplying and convolving Gaussian distributions will generate new Gaussian

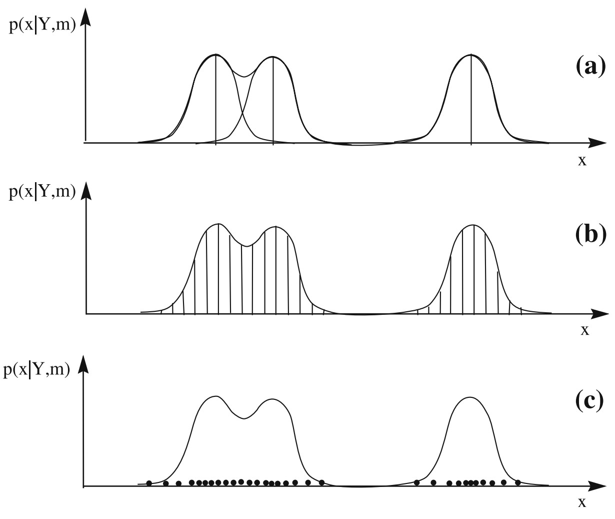

Fig. 1 Probability density approximations: (**a**) mixture of Gaussians, (**b**) piecewise approximation, (**c**) Monte Carlo approximation

distributions [5, 16, 25]. Multi-hypothesis Extended Kalman Filters falls into this class of filters. The conventional Extended Kalman Filter is unimodal, hence for multi-modal distributions the filter will fail. One possibility to deal with multi-modal distributions is to use a different Extended Kalman Filter to pursue each possible hypothesis about the robot pose. The combination of all Gaussian distributions together with the belief in each hypothesis can be interpreted as the global posterior probability distribution from a Bayesian point of view. In this approach, the integrals are not evaluated explicitly, but conceptually are replaced by summations.

- *Discretization of the state space*

  The *grid-based approach* divides the state space into regions and express the probability of being in each region [3, 9, 22]. The grid mesh defines an approximation of the conditional density given by a point mass value in each grid node position. This description induces an approximation of each probability integral by a finite sum over nonzero grid points values (see Fig. 1b).

  In this approach the posterior probability distribution is approximated by a set of point values on a grid

$$p(x_t|Y_t) \approx \sum_{i=1}^{N} p_i \delta\big(x_t - x_t^i\big) \tag{10}$$

  Each of these N grid points are equipped with a corresponding probability mass weight $p(x_t^i|Y_t)$, $k = 1, \ldots, N$. This technique substitutes the evaluation of the integrals in the Bayes' recursive expression by a sum of cells probabilities, and starts with a uniform probability distribution over the whole three dimensional state-space of the robot.

- *Monte Carlo approximation of the integrals*

  The Monte Carlo Localization methods are an alternative class of filters called sampling-based methods in which theoretical probability distributions on the state space are approximated by simulated random measurements [7, 17, 23]. In those methods the probability density function is represented by a set of N randomly sampled points called *particles* $S_t = \{x_t^i; i = 1, \dots, N\}$ (see Fig. 1c). Based on the samples it is possible to replace the infinite integrals with sums of weighted samples. In the Bayesian importance sampling this cloud is supposed to be independent and independently distributed draw from the proposal distribution yielding the following approximation

$$p(x_t|Y_t) \approx \sum_{i=1}^{N} w_t^i \delta\left(x_t - x_t^i\right) \tag{11}$$

where w_t^i are the normalized importance weights.

The Monte Carlo filtering technique main limitations comes from the high computational cost required and also from the relatively slow convergence of the algorithm.

4 Maximum a posteriori localization

From a maximum a posteriori point of view, the localization problem is basically an optimization problem, where the robot seeks to estimate the pose which maximizes the a posteriori probability density.

$$\begin{aligned}
\hat{x}_t^{\mathrm{MAP}} &= \arg\max_x p(x_t|Y_t) \\
&= \arg\max_x p(z_t|x_t, u_{t-1}, Y_{t-1}) p(x_t|x_{t-1}, u_{t-1}, Y_{t-1}) \\
&= \arg\max_x p(z_t|x_t) p(x_t|x_{t-1}, u_{t-1}) p(x_{t-1}|Y_{t-1}) \\
&= \arg\max_x \prod_{i=1}^{t} p(z_i|x_i) \prod_{i=1}^{t} p(x_i|x_{i-1}, u_{t-1}) p(x_0)
\end{aligned} \tag{12}$$

This expression requires to specify $p(x_t|x_{t-1}, u_{t-1})$ and $p(z_t|x_t)$. These distributions will be obtained from the mobile robot state space model.

The *maximum a priori* (MAP) estimate expression can be easily stated as an optimization problem in the presence of constraints in terms of the dynamical and observation models of the robot. The MAP estimate is the solution $\hat{x}_t$ to the following problem, subject to conditions (4).

$$\max_x \prod_{i=1}^{t} p_e(z_i|x_i) \prod_{i=1}^{t} p_v(x_i|x_{i-1}, u_{t-1}) p(x_0) \tag{13}$$

where p_e express the probability density function for the observation noise e, and p_v indicates the probability density function for the motion noise v. The expression 13 can be reformulated in an equivalent and convenient form by taking logarithms

$$\max_x \left[\sum_{i=1}^{t} \log p_e(z_i|x_i) + \sum_{i=1}^{t} \log p_v(x_i|x_{i-1}, u_{i-1}) + \log p(x_0) \right] \tag{14}$$

In general, the calculation of estimates for this optimization problem have no explicit analytical solutions for nonlinear and non-Gaussian models, and have to be iteratively solved to avoid the difficulties included in the optimization problem. These difficulties derives from the following aspects:

(1) It is highly nonlinear. Non-linearities due to motion and perception functions are propagated through the a posteriori robot pose probability density function.
(2) Environment symmetries make the objective function to maximize multi-modal. At initial stages the objective function admits a high number of solutions, even with the same maximum value. That happens in highly symmetric environments as typical offices buildings. The reason can be noticed in 14 where second term p_v is a constant in absence of robot motion and the third term $p(x_0)$ is also constant in absence of initial pose information. This lead the objective function $\max_x \quad \sum_{i=1}^{t} \log p_e(z_i|x_i)$ which only depends on observations and in highly regular environments has potentially multiple maxima.
(3) Another source of symmetries is originated by sensor limitations. The range and angular resolution of the sensor adds observational symmetries.

In order to solve 14 a set of candidate estimates have to be initially generated, maintained or pruned according to the new observation and motion information included in the objective function. The problem is simplified in case the initial probability distribution is Gaussian, because the problem becomes uni-modal and then it is possible to obtain, even analytically, an estimate (due to the problem can be converted in a quadratic minimization problem if non linear motion and observation models can be approximated by a linear Taylor series expansion about the current estimate $\hat{x}_t$). This situation lead us to the well known Extended Kalman Filter solution of the position tracking problem.

We will use the notation $f_0(x)$ to refer the objective function to maximize. The problem of finding an x that maximizes $f_0(x)$ among all x that satisfy the conditions $x_{t+1} = f(x_t, u_t) + v_t$ and $z_t = h(x_t) + e_t$ is limited to finding the optimal value between the set of all feasible points. A pose is feasible if satisfies the constraints $f()$ and $h()$. In the problem under consideration, there exists, at least at initial stages, multiple optimal values, then methods to solve the problem require to be able to manage a set of solutions. Bayesian methods use the *a posteriori* probability density function to do that, as was previously commented. The method proposed here uses a different approach, the idea is to maintain a set of N feasible solutions to the problem, and let this set evolve towards optimal values according to the observed motion and perception data and the constraints.

A second aspect required to solve in this optimization problem is how to obtain the optimal solutions at each time step. When f_0 is differentiable, a well known approach is based on the gradient vector of the objective function:

$$\frac{\partial f_0(x_t)}{\partial x_t} = 0 \tag{15}$$

Briefly, the aim of the estimation is to determine an estimate for each t, so that it defines a sequence of roots of the objective function equation that is consistent and asymptotically efficient. For linear estimation models in general, the function usually has a global maximum in the interior of the parameter space. In case the objective function is not linear, the gradient vector of the objective function $S(z_t, x_t) = \frac{\partial f_0(x_t)}{\partial x_t}$ can be approximated by a linear Taylor series expansion about the current iteration x_t^k for x_t. This gives

$$S(z_t, x_t) \approx S(z_t, x_t^k) - I(x_t^k, z_t)(x_t - x_t^k) \tag{16}$$

where $I(x_t^k, z_t)$ is the second-order partial derivative of the objective function with respect to the parameters vector to estimate x_t (the hessian matrix). A new fit x_t^k is obtained by taking it to be the zero of the right-hand side of 16. Hence

$$x_t^{k+1} = x_t^k + I^{-1}\big(x_t^k, z_t\big)S\big(z_t, x_t^k\big) \tag{17}$$

If the objective function $f_0(x) = p(z_t|x_t, Y_{t-1})$ is concave and unimodal, then the sequence of iterates $\{x_t^k\}$ converges to the maximum a posteriori estimate of x_t, in one step if the a posteriori probability density function is a quadratic function of x_t. When the objective function is not concave and unimodal, the Newton–Raphson method is not guaranteed to converge from an arbitrary starting value. Newton–Raphson method in its basic form can be used for position tracking but it can not be used easily in global localization problem.

In spite of the inability of using gradient-based methods to provide a reasonable approach to this problem, the idea of moving the estimate iteratively toward the most promising direction (indicated by the gradient) can be adopted. In global localization, there is a set of optimal and sub-optimal feasible solutions and we have multiple gradients. The method proposed in this work is to use the direction between each pose included in the sub-optimal set and the best pose contained in it as a kind of fictitious gradient to iterate each pose of the sub-optimal set of solutions.

4.1 Recursive formulation

The MAP estimate formulated as an optimization problem in the form $\max_x f_0(x_t)$ subject to conditions (4), is not practical from a computational point of view. In order to implement the global localization algorithm in a robot, a recursive formulation is required. If we observe the objective function $f_0(x_t)$, it can be expressed recursively in the following way:

$$\begin{aligned} f_0(x_t) &= \sum_{i=1}^{t} \log p_e(z_i|x_i) + \sum_{i=1}^{t} \log p_v(x_i|x_{i-1}, u_t) + \log p(x_0) \\ &= \log p_e(z_t|x_t) + \sum_{i=1}^{t-1} \log p_e(z_i|x_i) \\ &\quad + \log p_v(x_t|x_{t-1}, u_{t-1}, m) + \sum_{i=1}^{t-1} \log p_v(x_i|x_{i-1}, u_{t-1}) + \log p(x_0) \\ &= \log p_e(z_t|x_t) + \log p_v(x_t|x_{t-1}, u_{t-1}) + f_0(x_{t-1}) \end{aligned} \tag{18}$$

If we are able to solve the optimization problem at time $t-1$, we have a set of sub-optimal solutions which satisfy the optimization problem up to time $t-1$. And the MAP optimization problem can be reformulated as

$$\max_x \log p_v(z_t|x_t) + \log p_e(x_t|x_{t-1}, u_{t-1}) \tag{19}$$

subject to conditions (4) and $x_{t-1} = x_{t-1}^*$, where x_{t-1}^* is the x which maximize the MAP optimization problem up to $t-1$. Then solving 19 we will obtain a recursive version of the MAP estimate.

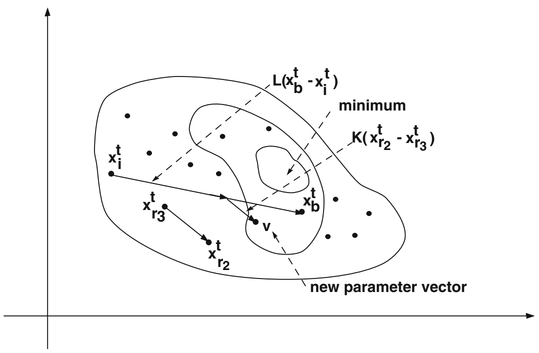

Fig. 2 New population member generation

In the following section an evolutive algorithm is proposed to obtain the evolutionary MAP estimate for the global localization problem according with the ideas introduced in this section.

5 ELF algoritm

The algorithm proposed to implement the evolutive filter is based on the differential evolution method proposed by Storn and Price [21] for global optimization problems over continuous spaces. The Evolutive Filter uses a parallel direct search method which utilizes n dimensional parameter vectors $x_i^k = (x_{i,1}^k, \ldots, x_{i,n}^k)^T$ to point each candidate solution i to the optimization problem at iteration k for a given time step t. This method utilizes N parameter vectors $\{x_i^k; i = 1, \ldots, N\}$ as a sub-optimal feasible solutions set (population) for each generation t of the optimization process.

The initial population is chosen randomly to cover the entire parameter space uniformly. In absence of a priori information the entire parameter space has the same probability of containing the optimum parameter vector, and a uniform probability distribution is assumed. The differential evolution filter generates new parameter vectors by adding the weighted difference vector between two population members to a third member. If the resulting vector yields a lower objective function value than a predetermined population member, the newly generated vector replaces the vector with which it was compared; otherwise, the old vector is retained. This basic idea is extended by perturbing an existing vector through the addition of one or more weighted difference vectors to it (see Fig. 2).

The perturbation scheme generates a variation v according to the following expression,

$$v = x_i^k + L\big(x_b^k - x_i^k\big) + F\big(x_{r_2}^k - x_{r_3}^k\big) \tag{20}$$

where x_i^k is the parameter vector to be perturbed at iteration k, x_b^k is the best parameter vector of the population at iteration k, $x_{r_2}^k$ and $x_{r_3}^k$ are parameter vectors chosen randomly from the population and are different from running index i. L and F are real and constant factors which controls the amplification of the differential variations $(x_b^k - x_i^k)$ and $(x_{r_2}^k - x_{r_3}^k)$. In our case and for simplicity reasons $L = F$ is adopted.

In order to increase the diversity of the new generation of parameter vectors, crossover is introduced. Denote by $u_i^k = (u_{i,1}^k, u_{i,2}^k, \dots, u_{i,n}^k)^T$ the new parameter vector with

$$u_{i,j}^k = \begin{cases} v_{i,j}^k & \text{if } p_{i,j}^k < \delta \\ x_{i,j}^k & \text{otherwise} \end{cases} \tag{21}$$

where $p_{i,j}^k$ is a randomly chosen value from the interval $[0, 1]$ for each parameter j of the population member i at step k and δ is the crossover probability and constitutes the crossover control variable. The random values $p_{i,j}^k$ are made anew for each trial vector i.

To decide whether or not vector u_i^k should become a member of generation $i+1$, the new vector is compared to x_i^k. If vector u_i^k yields a better value for the objective fitness function than x_i^k, then is replaced by u_i^k for the new generation; otherwise , the old value x_i^k is retained for the new generation. The general idea of the previous mechanism: mutation, crossover and selection are well known and can be found in literature [12].

5.1 Fitness function

According to the optimization problem under consideration

$$\max_x \sum_{i=1}^{t} \log p_e(z_i|x_i) + \sum_{i=1}^{t} \log p_v(x_i|x_{i-1}, u_{t-1}) + \log p(x_0) \tag{22}$$

subject to conditions (4). The natural choice for fitness function is the objective function.

$$f_0(x^t) = \sum_{i=1}^{t} \log p_e(z_i|x_i) + \sum_{i=1}^{t} \log p_v(x_i|x_{i-1}, u_{t-1}) + \log p(x_0) \tag{23}$$

This expression contains three probability densities. The perception model $p_e(z_i|x_i)$, the robot motion model $p_v(x_i|x_{i-1}, u_{t-1})$ and the initial a priori robot pose information $p(x_0)$.

The problem of computing $p_e(z_i|x_i)$ can be divided into three parts:

(1) The prediction of the value of the noise-free sensor assuming the robot pose estimate is $\hat{x}_t$, the sensor relative angle with respect to the robot axis is α_i and a given environment model m. Let $\hat{z}_{t,i} = h(\hat{x}_t, \alpha_i, m)$ denote this ideal predicted measurement. From a practical point of view, $\hat{z}_{t,i} = h(\hat{x}_t, \alpha_i, m)$ is computed using a ray tracing method. From an statistical point of view, and assuming the measurement error is Gaussian, this predicted measure will be the center of the Gaussian probability distribution of the expected distance measured by the α_i sensor when robot is located at x_t.

(2) The obtaining of the probability of observing $z_{t,i}$ data given the robot pose estimate. Assuming the measurement error $e_{t,i}$ is Gaussian, centered at $h(\hat{x}_t, \alpha_i, m)$ and with a σ_e standard deviation (that is $e_{t,i} \approx N(h(\hat{x}_t, \alpha_i, m), \sigma_e)$), then the probability of observing $z_{t,i}$ with sensor i can be expressed as,

$$p_e(z_{t,i}|\hat{x}_t) = \frac{1}{\left(2\pi\sigma_e^2\right)^{1/2}} e^{-1/2 \frac{(z_{t,i} - \hat{z}_{t,i})^2}{\sigma_e^2}} \tag{24}$$

(3) The integration of the individual sensor beam probabilities into a single probability value. Assuming conditional independence between the individual measurements:

$$p_e(z_t|\hat{x}_t) = \prod_{i=0}^{N_s} p(z_{t,i}|\hat{x}_t) = \prod_{i=0}^{N_s} \frac{1}{\left(2\pi\sigma_e^2\right)^{1/2}} e^{-1/2\frac{(z_{t,i}-\hat{z}_{t,i})^2}{\sigma_e^2}} \tag{25}$$

where N_s is the number of sensor observations.

The second probability required to calculate the objective function is $p_e(x_i|x_{i-1}, u_{t-1})$. The problem of computing $p_v(x_i|x_{i-1}, u_{t-1})$ can be divided into two parts:

(1) The prediction of the noise free value of the robot pose $\hat{x}_t$ assuming the robot pose estimate is $\hat{x}_t$ and the motion command at t is u_t. Let $\hat{x}_t = f(\hat{x}_{t-1}, u_{t-1})$ denote this ideal predicted state. From an statistical point of view, and assuming the motion error is Gaussian, this predicted measure will be the center of the Gaussian probability distribution of the expected distance when robot is located at $\hat{x}_t$.

(2) The obtaining of the probability of being at x_t given the robot pose estimate in $t, \hat{x}_{t-1}$ and the motion command u_{t-1}. Assuming the motion error v is Gaussian, centered at $\hat{x}_t$ and with a covariance matrix P (that is $v \approx N(f(\hat{x}_{t-1}, u_{t-1}), P)$, then the $p_v(x_i|x_{i-1}, u_{t-1})$ probability can be expressed as,

$$p_v(x_i|x_{i-1}, u_{t-1}) = \frac{1}{\sqrt{|P|(2\pi)^n}} e^{-1/2(x_i-\hat{x}_i)P^{-1}(x_i-\hat{x}_i)^T} \tag{26}$$

According to the recursive formulation (19), the objective function to optimize at iteration t is given by

$$f_0(x_t) = \log p_e(z_t|x_t) + \log p_v(x_t|x_{t-1}, u_{t-1}) \tag{27}$$

and introducing the expressions of p_v and p_e,

$$f_0(x_t) = \log \prod_{i=0}^{N_s} \left(2\pi\sigma_e^2\right)^{-1/2} e^{-\frac{(z_{t,i}-\hat{z}_{t,i})^2}{2\sigma_e^2}} + \log(|P|(2\pi)^n)^{-1/2} e^{-\frac{1}{2}(x_i-\hat{x}_i)P^{-1}(x_i-\hat{x}_i)^T} \tag{28}$$

$$f_0(x_t) = \sum_{i=0}^{N_s} \log\left(2\pi\sigma_e^2\right)^{-1/2} - \sum_{i=0}^{N_s} \frac{(z_{t,i}-\hat{z}_{t,i})^2}{2\sigma_e^2} + \log[(|P|(2\pi)^n)^{-1/2}] - \frac{1}{2}(x_i-\hat{x}_i)P^{-1}(x_i-\hat{x}_i)^T \tag{29}$$

which can be reduced to find the robot's pose to minimize the following function

$$f_0'(x_t) = \sum_{i=0}^{N_s} \frac{(z_{t,i}-\hat{z}_{t,i})^2}{2\sigma_e^2} + \frac{1}{2}(x_i-\hat{x}_i)P^{-1}(x_i-\hat{x}_i)^T \tag{30}$$

The differential evolutive filter will minimize iteratively the fitness function (30). The objective fitness function let us notice that the optimization considers the quadratic observation error and the quadratic pose error between the predicted pose and the pose under consideration weighted according its covariance matrix. Sometimes it can be interesting to modify the relative weight of terms in the right-hand side of (30), this can be done by multiplying by a scalar coefficient the second term. This coefficient modulates the importance given to the different information sources in the fitness function. It can be maintained constant or adaptively modified.

5.2 Analysis of the ELF algorithm

According with the previous ideas the algorithm has the following steps:

- *Step 1: Initialization*
  The initial set of solutions is calculated and the fitness value associated to each of the points in the state space is evaluated. In the most general case where no information about initial position is available, the initial set of pose solutions is obtained by drawing the robot poses according to a uniform probability distribution over the state space.
  The initial robot pose estimate is fixed to an initial value.
- *Step 2: Evolutive search*
  (a) For each element of the set of robot pose solutions, and according to the map, the expected sensor observations are obtained $\hat{z}_{t,i} = h(\hat{x}_t, \alpha_i, m), i = 1, \ldots, N_s$. The expected observations, the sensor observations, the robot pose estimate and the robot pose element are used to evaluate the loss function for each robot pose element in the set of solutions.
  (b) A new generation of perturbed robot pose solutions are generated according to the perturbation method exposed in Sect. 4. For each perturbed solution the expected observations are calculated and the loss function evaluated. If the perturbed solution results in a better loss function, this perturbed solution is selected for the following iteration, otherwise the original is maintained.
  (c) The crossover operator is applied to the resultant population (solutions set).
  (d) The robot pose element of the set with lower value of the loss function is marked as best robot pose estimate. Go to step 2b a given number of iterations.
- *Step 3: Updating*
  The best robot pose element of the population is used as the updated state estimate and then used in state transition model to predict the new state according to the odometry information.

$$\hat{x}_{t+1} = f(\hat{x}_t, u_t) \tag{31}$$

Then, the displacement is evaluated and the whole population is moved according to this displacement. Then go to step 2.

6 Experimental results

All experiments have been developed in an indoor environment—University laboratories, offices and corridors.

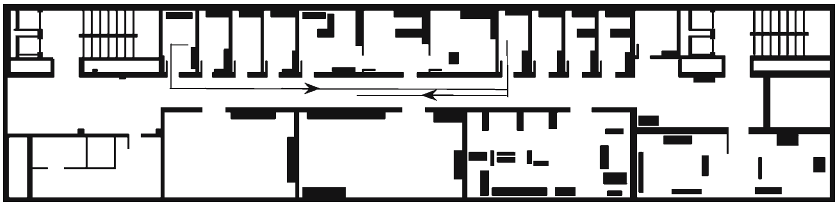

Fig. 3 Trajectory simulated for ELF algorithm

6.1 Convergence of the algorithm

One fundamental question about genetic algorithms is their convergence and their convergence rates: how quickly can the populations come to the individuals (robot pose solutions) with the highest fitness value? Some theoretical literature about genetic algorithm convergence to the global optimal (if it exists) shows that it is guarantied if the elitist strategy is used [13]. This results have been obtained modelling genetic algorithms as Markov chains and using the theory about the convergence rates of Markov chains to analyze the convergence of evolutionary algorithms. This strategy selects, for the following generation, the individual with the highest fitness value.

Three tests have been performed in this work where most important parameters have been modified to study their influence. In each test, 10 simulations of each different parameter have been done, and the medium values are presented in this section.

The trajectory simulated starts in an office, and continues along the central corridor toward the right side of the environment map (see Fig. 3). Then, the robot goes into the next seventh office, returns along the central corridor and finishes in the movement number 100.

First group of simulations has been carried out with a population of 500 individuals and 15 iterations for each movement and for different percentage of sensor error.

It has been considered that the algorithm converges to a solution, when all solutions in the population set are located in a ball of given radius around the best estimate (0.5 m in our experiments). Figure 4 shows the convergence speed of the Evolutive Filter as a function of the sensor error. The result shows the convergence rate reduction when the sensor error increases. In Fig. 4, the point where robot reaches the convergence is shown for each simulation.

A second test has been performed with the same population as in the previous test and a sensor error of 3%. In this case variation in the iteration number is studied.

Figure 5 shows that the convergence rate increases when the iteration number increases.

In the third group of simulation the constant variables are the sensor error (of 3%) and the iterations number (which is 15). The influence of the population size on the algorithm is studied and presented in Fig. 6.

In this case, the convergence speed increases with the number of individuals but above to 150 members the convergence speed does not improve. This is an important conclusion because a high number of individuals imply a time computing increment.

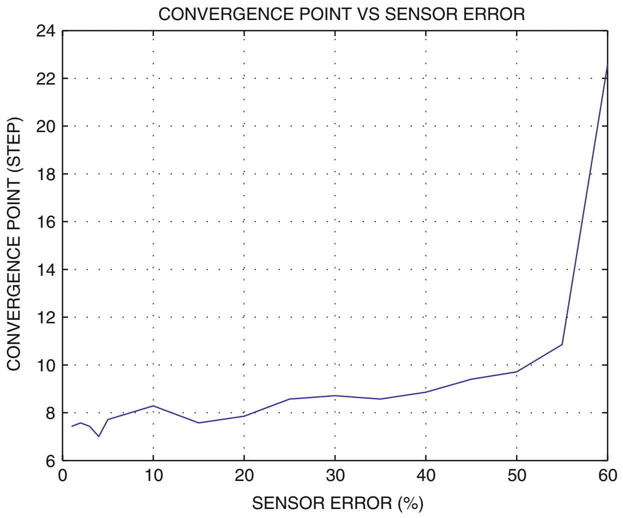

Fig. 4 Convergence speed as a function of the sensor error

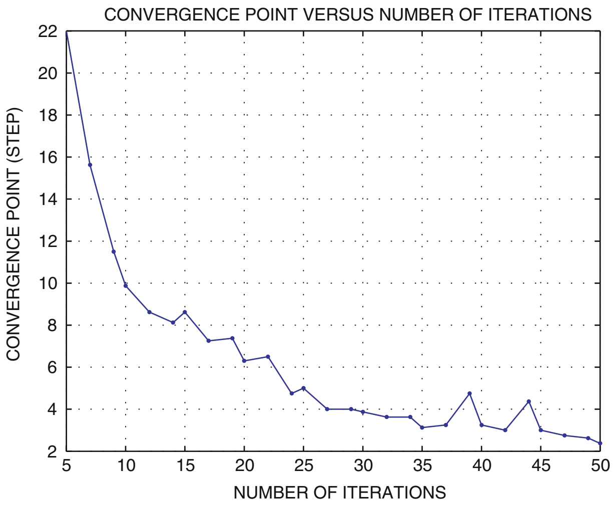

Fig. 5 Convergence speed as a function of the iterations number

As all the result suggest (Figs. 4, 5 and 6), the algorithm proposed is extremely efficient in terms of convergence speed. The behavior of the algorithm is clearly asymptotic.

6.2 Accuracy and robustness

One striking aspect is the low number of robot pose solutions required in the population. In Fig. 6, it can be noticed that convergence speed, that is the step which the convergence is reached, is standing under 9 steps for population sizes greater than 150 members.

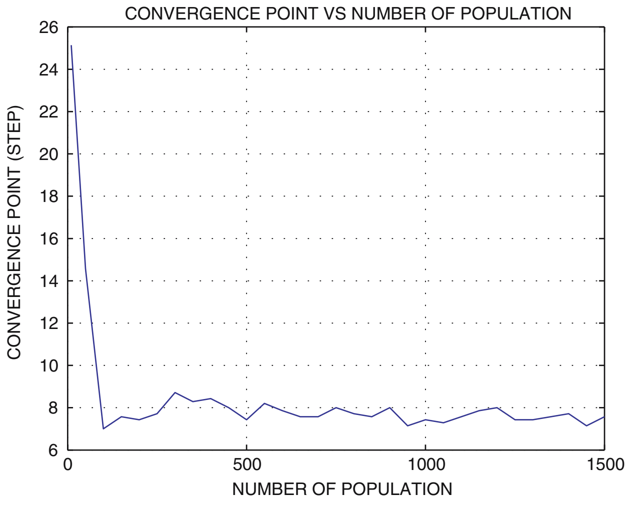

Fig. 6 Convergence speed as a function of the individuals number

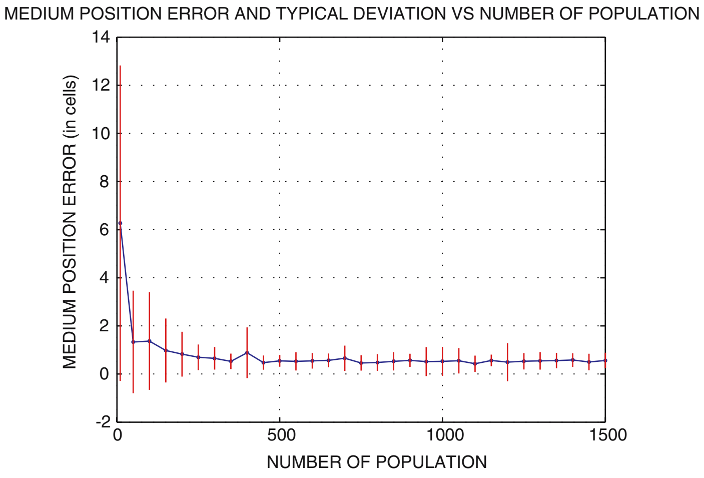

Fig. 7 Accuracy as a function of the population size

If we consider the effect of the population size on the accuracy of the algorithm (in Fig. 7), it can be noticed a minimum effect. This behavior is a consequence of the search nature of the ELF algorithm. This behavior differs completely from Monte Carlo results. As noticed by several authors ([8, 18]), the basic Monte Carlo filter performs poorly if the proposal distribution, which is used to generate samples, place not enough samples in regions where the desired posterior is large. This problem has practical importance because of time limitations existing in on-line applications. In this Fig. 7 a reduction of the position error is observed when the number of the members increases. Nevertheless, the reduction does not go on above 500 members. It must be also said that with 10 members in the population, different non convergence case was produced.

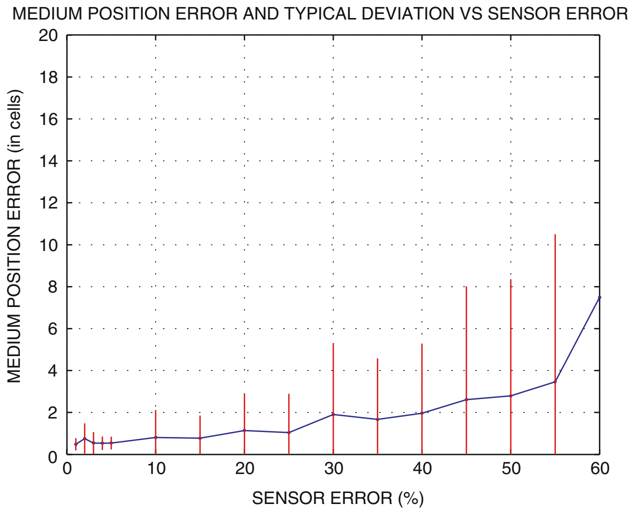

Fig. 8 Error of ELF (and typical deviation) as a function of the sensor noise

According these results, it can be noticed that 500 is an appropriate population size for the environment under consideration.The size of the environment under consideration is approximately 800 m^2, then only 0.62 elements per square meter is required. An interesting comparison with other particle or sampling based methods like Monte Carlo can be done. Jensfelt tests with a Monte Carlo Localization method [15] shows that the value of *N* has a critical effect on the robustness.

In Fig. 8, the medium position error and the typical deviation versus sensor error (in %) is presented. Medium values have been performed with 10 experiments for each simulation. Figure 8 shows a continuous increment in the medium error and its typical deviation when the sensor error increases. It must be highlighted that under a 10% sensor error, the medium position error is less than one cell.

Figure 8 shows the performance of the ELF under different sensor noise levels. The ELF algorithm exhibits a relatively constant average error less than 5 cm until a sensor noise level of 30%, but even with higher noise levels (50%) is able to localize the robot. This robustness to sensor noise is equivalent to Monte Carlo results shown in [23], but the accuracy for similar noise levels is higher. In those tests with Monte Carlo Localization, the error average is between 20 cm and 30 cm until a 10% of sensor noise ratio and then it degrades gracefully.

To finish, the error of a typical deviation of the ELF algorithm versus iteration number is presented in Fig. 9. The waited theoretical shape for this curve is like a bathtub, due to the hyperconvergence caused by the genetic algorithm for a high number of iterations. Nevertheless this has not been able to be observed in the experiments. Up to 10 iterations there are not an improvement in the mean position error. From 45 iterations, an error increment appear to be observed.

6.3 Complexity and computational cost

A final issue addressed in our experiments concerns the running time of ELF algorithm. The absolute time depends on several factors: the computer platform, the

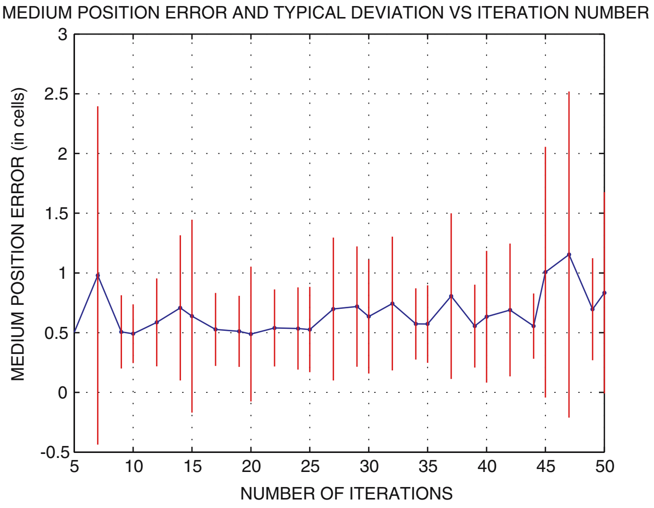

Fig. 9 Accuracy as a function of the iteration number

Table 1 Computation time for a fixed number of iterations

Population/Iterations	Time (ms)
1,000/15	1,703
500/15	859
400/15	704
300/15	520
200/15	359
100/15	180
75/15	141
50/15	94

observation prediction model and the sensor data, and the population and iterations number. This section illustrates the requirements of ELF. All results reported here were obtained in a PC with an Athlon XP 1800+ processor (Table 1).

Table 1 shows the time required for ELF to update the estimate value using 60 range laser data, 15 iterations and different population sizes. Table 2 shows the time required for ELF algorithm to update the robot pose estimate using a fixed population size and changing the iterations number. In both cases the behavior of the algorithm is completely linear. The complexity of the algorithm is then $O(N.M)$ where N is the population size and M is the iteration number. In our experimental localization tests, in a test area of 900 m^2 the best results have been obtained with an initial population of 500 elements and 15 iterations. Once the population converge into a ball of 1 meter radius, the population required to keep the robot localized decrease considerably. In our experiments 50 elements are sufficient. This result shows that even in the initial steps of the ELF algorithms the computational cost is moderate.

A critical point in any global localization algorithm is the variation in the computational requirements with the environment dimensions. EFL algorithm uses a very low samples density, but at initial stages the algorithm have to evaluate the whole environment map. Then the initial number of samples is proportional to the

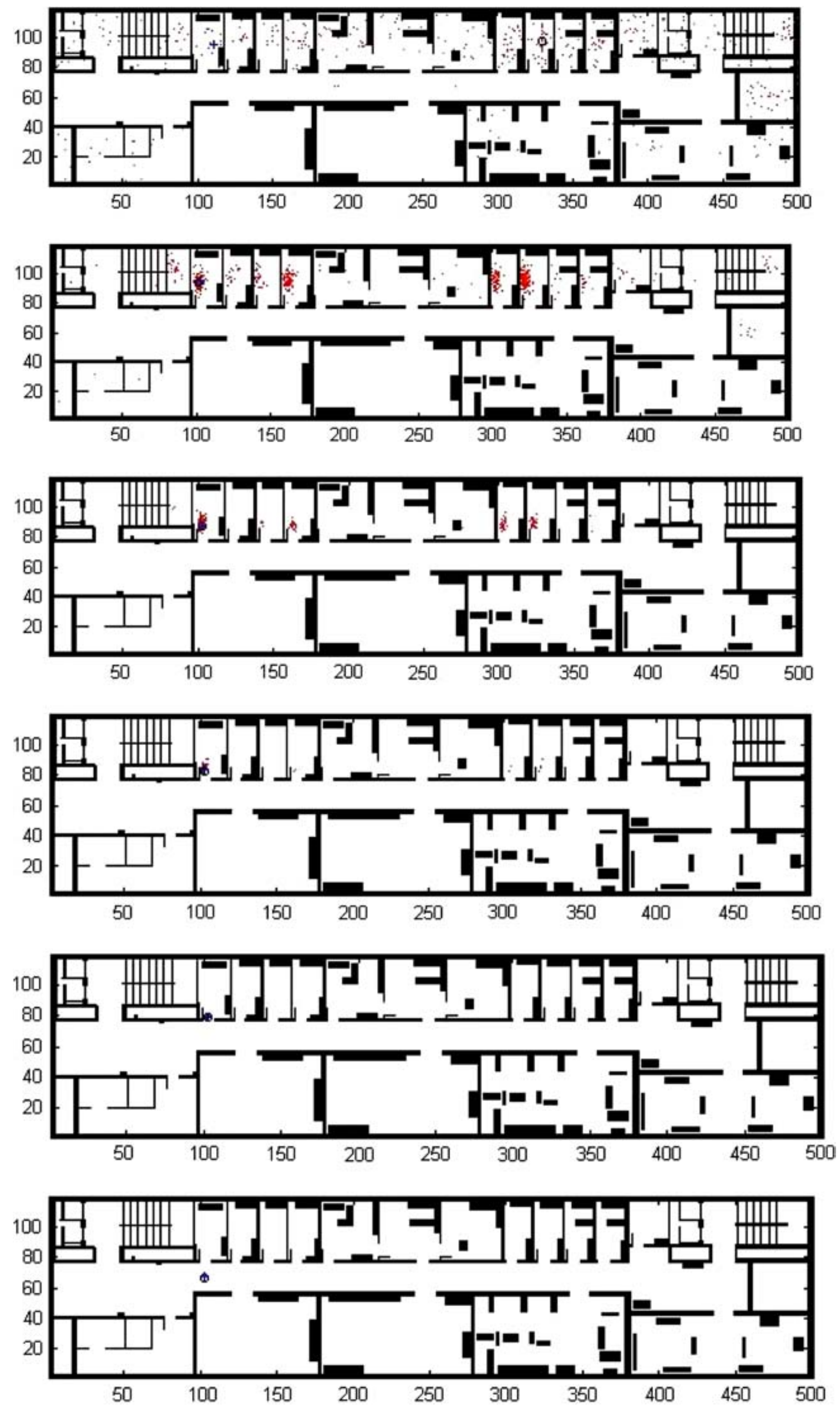

Fig. 10 Convergence evolution for 100 movements in positions: 1, 3, 5, 6, 7, 10. Convergence final is observed to be placed after movement number 10

Table 2 Computation time for a fixed population of 500 elements

Population/Iter.	Time(ms)
500/5	328
500/10	641
500/15	859
500/20	1,125
500/25	1,407

environment area. In the experiments done in our laboratory test site, to cope with a $900\,\mathrm{m}^2$ area the algorithms requires 500 samples and for an area of $1,800\,\mathrm{m}^2$ the samples required are $1,000$.

Figure 10 shows a typical convergence process of the Evolutionary Localization Filter for a population set of 1,000 individuals and 15 iterations of the Evolutionary Filter in each cycle. It can be noticed that the population individuals tend to concentrate in highly similar areas at the initial robot pose. After some cycles the algorithm estimate (circle) reaches the true robot pose (cross) and the population set converges to the true robot pose.

7 Related work

Kalman filtering algorithms usually do not use raw sensor data for localization. Instead, they extract features from which robot poses can be estimated. Different approaches can be found in the literature to extract features, such as points, lines, corners, etc. Using features instead of the raw sensor data can be loss-free if the features contain sufficient statistical information relative to the problem of estimating robot poses. In practice, there is no guarantee of sufficiency, and significant information may be lost when extracting features from raw sensor data.

In spite of these limitations, multi-hypothesis Kalman filters have been applied successfully to global localization problems and position tracking. Localization algorithms based on the multi-hypothesis Kalman filter represent probabilities using mixtures of Gaussians. To calculate the covariance matrices of the individual Gaussian mixture components, the Kalman filtering approach linearizes the motion model and the observation model. Noise in sensor measurement and robot motion is assumed Gaussian. The number of mixture components and the relative weight apply heuristic rules for terminating low weight Gaussians and to creating new ones. The number of hypothesis required depends on the symmetries and ambiguities of the environment perceived. These hypothesis, besides the integration of new sensor data by mean of Kalman equations requires to verify, to eliminate and sometimes to split up into several offspring hypothesis. The number of hypothesis can be high, in a typical indoor environment of 150 square meters [1] uses 72 hypothesis at initial stages of the algorithm which are progressively pruned. The hypothesis generation takes four times the time required for hypothesis tracking which limits the advantage of the computational efficiency of Kalman filtering. In any case, the number of potential hypothesis grows rapidly with the environment area and so does the hypothesis generation cost.

In grid-based approaches, the environment is decomposed into a number of regular cells (there exists versions with non regular decompositions whose size and location depends on the structure of the environment). In case of regular space state decompositions, the size of the grid is very big and grows very fast (due to robot pose is a three dimensional problem the number of cells required can be various millions even for moderate size environments). These methods use raw sensor data to update the probability distribution by means of the Bayes' rule. The disadvantage of grid-based localization methods lies in the huge number of grid-cells which have to be updated. A small-size environment of $10 \times 10\,\mathrm{m}$, with a cell size of 10×10 centimeter and 2° of angular resolution the state space is discretized in $1,800,000$ states. Each of these states has to be updated for any incoming sensor data and motion command executed by the mobile robot. In order to deal with such huge state spaces Fox applies a fast model for proximity sensors and a technique for selectively updating the probability state focused only on the relevant part of the space.

Monte Carlo based filters represent the probability distribution as a set of samples, and the most probable locations are represented by a high concentration of the weighted samples in those areas. These samples have to be carefully weighted, updated and re-distributed to cope the location ambiguities from environment or sensing. The ability of these techniques to properly react to sensor and motion information is due to the quantity of samples and a distribution strategy which must be appropriately chosen. The effectiveness of the method depends on the number of samples used to model the probability distribution, and is critical at initial stages of the algorithm where probability distribution is spread along the environment promising areas. These approaches have shown their efficiency to solve the problem, but the computational burden of this technique limits their practical applications. Some adaptive schemes ([15, 23]) have improved considerably the number of samples required to obtain reliable results, but the number of samples required remains considerable until the algorithm converge to the correct robot pose. Besides, the number of iterations at each cycle is also elevated, Jensfelt tests require around 40 iterations to converge to the adequate posterior probability distribution. Thrun has evaluated the number of samples required and according to his test good results has been obtained within the range between $1,000$ and $5,000$ samples. This number of sample restricts the applications of this solution to off line problems. An interesting possibility in this technique consist on dynamically adapt the number of samples to the probability distribution situation. This adaptation allows for a reduction of the number of samples (100 can be enough) when the probability distribution is clearly concentrated on one hypothesis.

As noticed by several authors [18], the basic particle filter performs poorly if the proposal distribution, which is used to generate samples, places not enough samples in regions where the desired posterior probability is large. This problem has practical importance in the context of Monte Carlo localization. A solution adopted by some researchers consist on the addition of random samples into the posterior and the generation of samples at locations that are consistent with the sensor readings [20].

8 Conclusion

To the best of our knowledge, the evolutionary localization filter proposed here is new. The ELF algorithm presented in this article can handle arbitrary noise models and is capable of using raw sensor data (e.g., laser range data) for global localization.

The Markov chain nature of evolutionary algorithms is exploited by introducing in the loss function the sensor error innovation together with motion error innovation. In the method proposed here each individual in the evolutive algorithm will represent a possible solution to the localization problem and the value of the loss function represent the error to explain the perceptual and motion data. The search of this solution is done stochastically employing an evolutive search technique.

The evolutive optimization technique constitutes a probabilistic search method that avoids derivatives. The use of derivatives present two types of problems:

- Causes strong numerical oscillations when noise to signal ratio is high.
- Requires differentiable functions, because otherwise the derivatives can be discontinuous or even do not exist.

The set of solutions (the population) is modified according to the natural evolution mechanism: selection and crossover, in a recursive loop. Each loop iteration corresponds to one generation, and represent the set of solutions (population) at this moment. The selection operator tries to improve the medium quality of the set of solutions by giving higher probability to be copied to next generation to the best solutions. This operator has a substantial significance because it focuses on the search of best solutions in the most promising regions of the state space. The quality of an individual solution is measured by means of the fitness function. It has been demonstrated [10, 11], that Genetic Algorithms operating in restricted areas of the solution space can be a fast optimization method for time-varying, non linear and non differentiable functions.

This article introduced a new mobile robot localization technique, called ELF. ELF possesses a range of advantages over previous localization algorithms capable of global localization from ambiguously interpretable data information:

(1) The algorithm can accommodate arbitrary non-linear system dynamics, sensor characteristics and non-Gaussian noise. By introducing in the fitness function the sensor innovation together with the motion innovation, and due to the Markov Chain behavior of the evolutive search algorithm the set of particles evolves gradually along the most probable environment areas.
(2) Since the set of solutions does not try to approximate posterior density distributions, it does not require any assumptions on the shape of the posterior density as parametric approaches do.
(3) Evolutive filter focus computational resources in the most relevant areas, by addressing the set of solutions to the most interesting areas according to the fitness function obtained.
(4) Tests indicate that the number of tentative solutions required in the evolving set is lower than those required in particle filters, and similarly to those filters the evolving set can be reduced when the algorithm has converged to a reduced area around the best estimate.
(5) The size of the minimum solution's set required to guaranty the convergence of the evolutive filter to the true solution is low.
(6) The algorithm is easy to implement, and the computational cost makes it able to operate on line even in relatively big areas.
(7) Due to the stochastic nature of the algorithm search of the best robot pose estimate the algorithm is able to cope a high level of sensor noise with low degradation of the estimation results.

References

Arras, K., Castellanos, J.A., and Siegwart, R.: Feature-based multi-hypothesis localization and tracking for mobile robots using geometric constraints. In: Proc. of the Int. Conference on Robotics and Automation ICRA-02, pp. 1371–1377. Washington DC, USA (2002)

Austin, D.J., Jensfelt, P.: Using multiple Gaussian hypotheses to represent probability distributions for mobile robot localization. In: Proc. of the Int. Conference on Robotics and Automation ICRA-00, pp. 1036–1041. San Francisco, CA, USA (2000)

Burgard, W., Fox, D., Henning, D., Schmidt, T.: Estimating the absolute position of a mobile robot using position probability drids. In: Proc. of the National Conference on Artificial Intelligence AAAI-96, pp. 896–901. Potland, Oregon, USA (1996)

Cox, I.J.: Blanche-An Experiment in guidance and navigation of an autonomous robot vehicle. IEEE Trans. Robotics Autom. **7**, 193–204 (1991)

Cox, I.J., Leonard, J.J.: Modeling a dynamic environment using a Bayesian multi hypothesis approach. Artif. Int. **66**, 311–344 (1994)

Crowley, J.L.: World modelling and position estimation for a mobile robot using ultrasonic ranging. In: IEEE International Conference on Robotics and Automation. Scottsdale, AZ (1989)

Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte Carlo Localization for Mobile Robots. In: Proceedings of the 1999 International Conference on Robotics and Automation, pp. 1322–1328 (1999)

Doucet, A.: On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/FINFENG/TR 31', Cambridge University, Dept of Engineering, Cambridge, UK (1998)

Fox, D., Burgard, W., Thrun, S.: Markov localization for mobile robots in dynamic environments. J. Artif. Int. Res. **11**, 391–427 (1999)

Garrido, S., Moreno, L., Salichs, M.A.: Non linear on line identification of dynamic systems with restricted genetic optimization. In: Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing, EUFIT, pp. 423–428 (1998)

Garrido, S., Moreno, L.: Learning adaptive parameters with restricted genetic optimization. In: Bio-inspired Applications of Connectionism: 6th International Work-Conference on Artificial and Natural Neural Networks, IWANN2001, pp. 612–620. Springer-Verlag (2001)

Goldberg, D.E.: Genetic algorithm in Search, Optimization, and Machine Learning. Addison Wesley Publishing Company (1989)

He, J., Kang, L.: On the convergence rates of genetic algorithms. Theor Comput. Sci. **229**, 23–39 (1999)

Jensfelt, P., Wijk, O., Austin, D.J., Andersson, M.: Experiments on augmenting condensation for mobile robot localization. In: Proc. of the Int. Conference on Robotics and Automation ICRA-00, pp. 2518–2524. San Francisco, CA, USA (2000)

Jensfelt, P.: Approaches to mobile robot localization in indoor environments. Doctoral Thesis, Royal Institute of Technology, Sweden (2001)

Jensfelt, P., Kristensen, S.: Active global localisation for a mobile robot using multiple hypothesis tracking. In: Proc. IJCAI Workshop on Reasoning with Uncertainty in Robot Navigation, pp. 13–22. Stockholm, Sweden (1999)

Kalos, M.H., Whitlock, P.A.: Monte Carlo Methods, Vol. I: Basics. John Wiley and Sons (1986)

Liu, J., Chen, R.: Sequential Monte Carlo methods for dynamic systems. J. Amer. Statis. Assoc. **93**, 1032–1044 (1998)

Leonard, J.J., Durrant-White H.F.: Directed Sonar Sensing for Mobile Robot Navigation. Kluwer Academic Publishers (1992)

Lenser, S., Veloso, M.: Sensor resetting localization for poorly modelled mobile robots. In: Proc. IEEE International Conference on Robotics and Automation (ICRA-2000). San Francisco, CA (2000)

Storn, R., Price, K.: Differential Evolution–A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, March (1995)

Thrun, S.: Bayesian landmark learning for mobile robot localization. Mach. Learn. **33**(1), 41–76 (1998)

Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Robust Monte Carlo localization for mobile robots. Artif. Int. **128**, 99–141 (2001)

Reuter, J.: Mobile robot self-localization using PDAB. In: Proc. IEEE International Conference on Robotics and Automation (ICRA-2000). San Francisco, CA (2000)

Roumeliotis, S.I., Bekey, G.A.: Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization. In: Proc. IEEE International Conference on Robotics and Automation (ICRA-2000). pp. 2985–2992. San Francisco (2000)