2011 IEEE International Conference on Fuzzy Systems
June 27-30, 2011, Taipei, Taiwan

Differential Evolution to Enhance Localization of Mobile Robots

Michał Lisowski
Department of Management Engineering
Technical University of Denmark
2800 Kgs. Lyngby, Denmark
michlis@gmail.com

Zhun Fan
Department of Management Engineering
Technical University of Denmark
2800 Kgs. Lyngby, Denmark
zhfa@man.dtu.dk

Ole Ravn
Department of Electronic Engineering
Technical University of Denmark
2800 Kgs. Lyngby, Denmark
or@elektro.dtu.dk

***Abstract*—This paper focuses on the mobile robot localization problems: pose tracking, global localization and robot kidnap. Differential Evolution (DE) applied to extend Monte Carlo Localization (MCL) was investigated to better solve localization problem by increasing localization reliability and speed. In addition, a novel mechanism for effective robot kidnap detection was proposed. Experiments were performed using computer simulations based on the odometer data and laser range finder measurements collected in advance by a robot in real-life. Experimental results showed that integrating DE enables MCL to provide more accurate robot pose estimations in shorter time while using fewer particles.**

Keywords-Monte Carlo Localization; Differential Evolution; particle filter; mobile robot

I. INTRODUCTION

With the advances in intelligent sensing and control, in the near future, robots could take over more and more tasks and provide valuable services. Potential applications are diverse, from cleaning and transportation through enhanced communication (e.g. telepresence) and entertainment to security and military use. Unfortunately, nowadays it is still more likely to find mobile robot in structured environment, such as in a research laboratory or in a factory, rather than at home or at the office. Even though a significant progress in mobile robotics field has been made in the last two decades, the levels of autonomy presented by most of mobile robots are still limited, and the interaction and the cooperation between humans and robots are still challenges. To achieve this state, when mobile robots are commonly present in our everyday lives, one of the first remaining problems is localization, which could involve three sub-problems: global localization, pose tracking and kidnap problems.

There is a number of ways to approach the stated problem, which potentially may yield satisfactory solutions. This paper is dedicated to particle filters and evolutionary algorithms, particularly Monte Carlo Localization (MCL) and Differential Evolution (DE). Although both methods are mature and have long history, they were applied together to mobile robot localization problem only once by Moreno et al. [1]. Combining both methods has the potential of yielding a more effective algorithm. To our best knowledge, there was no other evidence in the literature of research in mobile robotics field regarding specifically MCL and DE. Moreno et al. presented an evolutive localization algorithm for mobile robots combining DE and MCL. Algorithm implementation is based on occupancy grid maps and laser range finder sensor.

Thrun et al. [2] presented a comprehensive paper on MCL, which is a family of MCL algorithms including Mixture-MCL. This algorithm combines classic MCL, where poses are predicted based on robots motion, with dual MCL, where information from sensors is used to predict robot poses. One of the interesting observations on classic MCL limitations are that the algorithm performs poorly if not enough particles are placed close to the true robot pose. This requires proposal distribution, from which particles are drawn, to correspond well to the true distribution of robot poses. Recently, Blanco et al. [3] proposed an optimal filtering algorithm, which dynamically generates the minimum number of particles that best represent the true distribution.

An interesting attempt to combine MCL with evolutionary mechanisms is presented by Ronghua and Bingrong [4]. The main focus in this paper is the problem of premature convergence when using MCL in highly symmetric environments. Another solution to mobile robot localization problem based on MCL and evolutionary approach is presented by Gasparri et al. [5]. The key concept of the algorithm presented in this paper is a dynamic clusterization method, which allows applying the evolutionary filter locally in each particle subset. This approach results in parallel exploration of the environment by a number of subsets.

Although not based on MCL, an interesting genetic algorithm for mobile robot localization is presented by Moreno et al. [6]. It also applies evolutionary filter only locally, but uses Extended Kalman Filter (EKF) instead of MCL to provide an estimate of robots pose. The proposed algorithm was tested indoor using a mobile robot equipped with 24 ultrasonic sensors, and demonstrated remarkable improvement in the results and was less computationally expensive than MCL. In this paper, we propose an algorithm combining MCL and DE that can improve the effectiveness of the algorithm, and discusses the performance improvement compared with the standard MCL method.

978-1-4244-7317-5/11/$26.00 ©2011 IEEE

The remaining part of this paper is organized as follows. In Section II, the theory of Monte Carlo Localization and Differential Evolution, and recent solutions to mobile robot localization problem using evolutionary approach and particle filters are introduced. An algorithm combining MCL with DE is presented and discussed. In Section III, an improved algorithm based on MCL and DE is presented and explained. Section IV gives experimental results on comparison of DEMCL with MCL in terms of robot pose tracking, and The proposed modifications to DEMCL algorithm is also evaluated in terms of robot kidnap problem and global localization. Finally, conclusions are presented in Section V.

II. Differential Evolution Monte Carlo Localization (DEMCL)

A. Concepts of Monte Carlo Localization and Differential Evolution

Monte Carlo Localization (MCL) represents a sampling-based class of filters in which theoretical probability distribution on the state space is approximated by a finite set of samples

$$X_t = \{x_t^p; p = 1, 2, ...P\} \tag{1}$$

called particles, for each time-step t. Every particle is a single point in the state space and represents a hypothesis (belief) about the true state.

In robot localization, we are interested in estimating the current state of the robot x_t. Typically, two-dimensional case is considered where $x_t = [x, y, \theta]^T$, i.e. x_t. represents robot position in XY-coordinates and orientation. State estimation is based on the input data Y_t. Here, Y_t represents all the data gathered from time-step 0 until t. In mobile robotics, typically two types of data are used: sensor readings $Z_t = \{z_0, ..., z_t\}$ and motion readings (odometry) $U_t = \{u_0, ..., u_t\}$. Often in practice both types of data arrive alternately i.e. $Y_t = \{z_t, u_{t-1}, z_{t-1}, u_{t-2}, ..., z_1, u_0, z_0\}$. Since odometry data u_t represents robots movement (motion command) from state x_t to x_{t+1}, at time-step t the most recent available odometry data is u_{t-1}, as indicated in Figure 1. The MCL algorithm providing recursive state estimation of the robot pose is illustrated in Algorithm 1.

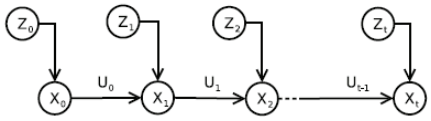

Figure 1. Robot state transformation and input data diagram.

Algorithm 1 MCL(X_{t-1}, u_t, z_t, m)

1: $X'_t = X_t = \phi$
2: **for** $p = 1$ to P **do**
3: x_t^p = **sample_motion_model**(u_t, x_{t-1}^p)
4: w_t^p = **measurement_model**(z_t, x_t^p, m)
5: $X'_t = X'_t + \langle x_t^p, w_t^p \rangle$
6: **end for**
7: **for** $p = 1$ to P **do**
8: draw $p \in [1, \ldots, P]$ with probability $\propto w_t^p$
9: add x_t^p to X_t
10: **end for**
11: **return** X_t

Differential Evolution was proposed by Storn and Price [7] for global optimization problems over continuous spaces. As any other evolutionary algorithm, DE is working with a number of potential solutions, often called a population, rather than a single one. The only requirement for DE is to provide initial population and define an objective function. It allows comparing different solutions in terms of fitness and thus is often referred to as the fitness function. Little requirements result in a wide range of optimization problems which can be addressed with DE. Even the initial requirement for continuous spaces is not valid any more, since in the recent years DE has been extended and successfully applied also to permutation-based combinatorial problems [8].

From the mathematical perspective, DE uses NP parameter vectors

$$x_i^g, i \in [0, NP-1], g \in [0, g_{\max}] \tag{2}$$

forming a population

$$P_g = [x_i^g; i = 1, 2, ..., NP] \tag{3}$$

of potential solutions. Generation index g is an integer equal or greater than zero and limited by g_{max}. P_0 corresponds to initial population, while P_1 represents population after first iteration (step) of the evolutive filter.

Additionally, each parameter vector *i*

$$x_i^g = [x_{i,1}^g, ..., x_{i,D}^g]^T \tag{4}$$

represents a single point in the D-dimensional problem space at generation g.

B. Combination of Monte Carlo Localization and Differential Evolution

Combining MCL with DE is a natural task, because the core concepts of both algorithms are very similar as presented in Table I.

TABLE I. Comparison of core concepts of MCL (particle and set) and DE (individual and population).

MCL	***DE***
$x_t^p = [x, y, \theta]^T$	$x_i^g = [x_{i,1}^g, ..., x_{i,D}^g]^T$
$X_t = \{x_t^p; p = 1, 2, ...P\}$	$P_g = [x_i^g; i = 1, 2, ..., NP]$

If the dimension D of DE parameter vector x_i^g (individual) is set to 3, then it is equivalent to particle x_t i.e. $x_{i,1}^g = x$, $x_{i,2}^g = y$ and $x_{i,3}^g = \theta$. Additionally, particle set X_t becomes equivalent to population P_g, when the number of particles P is equal to the number of parameter vectors NP.

GAs implementation requires a solution space representation and a fitness function. In this case, the most simple and intuitive solution space is $x_t = (x, y, \theta)$, because it can be directly used to represent both particles and individuals. The most obvious choice for fitness function is the measurement model, because it evaluates individual particles in terms of solving the localization problem. Therefore particle weight in MCL can be also considered as a fitness value of individual in DE.

On the algorithm level, DE can successfully replace resampling process in MCL due to crossover and mutation operators [9]. In this way, new algorithm (Algorithm 2) combines DE with MCL and will be referred to as the Differential Evolution Monte Carlo Localization (DEMCL).

As it can be seen in Algorithm 2, each step of DEMCL algorithm starts by updating particles with the most recent robot movement and assigning weights according to the measurement model. This is exactly the same as in MCL, but instead of re-sampling particles are now filtered using DE. In case of Algorithm 2, evolution is performed in a loop for a given number of generations g_{max}. Last generation P_{gmax} becomes the particle set X_t, which estimates robot pose at time t.

Algorithm 2 DEMCL(X_{t-1}, u_t, z_t,m, F,CR, g_{max}))

$X'_t = X_t = \phi$
for $p = 1$ to P do
  $x_t^p = \text{sample\_motion\_model}(u_t, x_{t-1}^p)$
  $w_t^p = \text{measurement\_model}(z_t, x_t^p, m)$
  $X'_t = X'_t + \langle x_t^p, w_t^p \rangle$
end for
$g = 0, P_0 = X'_t$
while $g < g_{max}$ do
  for $p = 1$ to P do
    $v^p = x_t^{r1} + F \cdot (x_t^{r2} - x_t^{r3}),\ r1, r2, r3 \in [1, P],\ F > 0$
    $u^p = \begin{cases} v_j^p & \text{if } (p_{p,j} \le CR \text{ or } j = j_{rand}) \\ x_{t,j}^p & \text{otherwise} \end{cases} \quad j = 1, 2, 3$
    $w_u = \text{measurement\_model}(z_t, u^p, m)$
    if $w_u > w_t^p$ then
      $P_{g+1} = P_{g+1} + \langle u^p, w_u \rangle$
    else
      $P_{g+1} = P_{g+1} + \langle x_t^p, w_t^p \rangle$
    end if
  end for
  $g = g + 1$
end while
$X_t = P_{g-1}$
return X_t

DE mutation variant used in Algorithm 2 is DE/rand/1/bin with F = 0.5 and CR = 0.7 (which corresponds to ~80% crossover probability).

III. IMPROVED DEMCL

A. Motivation

As demonstrated in the previous chapter, DEMCL is very suitable for solving both global localization and pose tracking problems. However, in the current form it cannot address robot kidnap problem. The main reason is that once population is converged the difference vectors in mutation are small, so evolutionary search is limited only to particles neighborhood. In other words, when all particles are grouped together, new candidate particles are generated by DE only locally. As a result, once the robot is kidnapped to a distant location, the probability of generating a new particle close to its true pose is practically zero and therefore it is almost impossible to recover from the kidnap.

It is evident that in order to solve the kidnap problem it is necessary to modify DEMCL algorithm.

B. Method description

The most straightforward solution to kidnap problem is to extend DE search well beyond the converged population by using high scaling factor F in the mutation operator. Typical F range is (0-1) and in our experiments F=0.5, but theoretically F ∈ R+. When using high F values, mutated particles are more widely spread in the map and the probability of generating a particle close to the true robot pose after kidnap becomes larger. However, the problem is how to select correct F value. There are two methods for randomizing F via selecting a PDF and deciding how often to draw F, either for each parameter in the vector (jitter) or for every vector (dirther), which require a range of values instead of a single point. However, if the F range is not proper, mutated particles could be often outside the map and this approach would be very inefficient, just as adding random particles to the set.

Taking into account good global localization results of DEMCL, a more efficient solution to kidnap problem would be to reinitialize global localization search once the robot is kidnapped. We propose to modify DEMCL so that the population can be reset by redistributing all particles randomly and uniformly in the map. This approach requires a method to detect the kidnap. We propose following rules to signal when population resetting should be performed:

1. If the average particle weight has decreased recently compared to the long-term average.

2. If the particle set is converged but the long-term average particle weight is low.

3. If the particle set is old and still not converged.

Rule 1 First rule originates from the sensor resetting technique because it also monitors the long term (w_{slow}) and short term (w_{fast}) averages of particles weights using exponential filters. The main purpose of this rule is to signal when the particle set starts to diverge from the true pose distribution. For this rule, the probability of reset is defined as

$$pr_1 = \min(0, 1 - \frac{w_{fast}}{w_{slow}}) \quad (5)$$

where

$$w_{slow} = w'_{slow} + \alpha_{slow}(w_{avg} - w'_{slow}), \alpha_{slow} = 0.2, w_{slow} \in [0,1]$$
$$w_{fast} = w'_{fast} + \alpha_{fast}(w_{avg} - w'_{fast}), \alpha_{fast} = 0.95, w_{fast} \in [0,1]$$

Rule 2 Second rule is designed to assist the first one when the particle set is converged to a wrong pose. In such situations it is reasonable to expect low w_{slow} values, so the probability of reset is defined in this case as

$$pr_2 = \min(0, \alpha_1 - w_{slow}) \quad ()$$

where $1 \in (0, 1)$ is the upper limit of unacceptable w_{slow} values. If w_{slow} 1 than pr2 = 0 and population reset is not triggered by this rule. Parameter1 is introduced in this rule to serve as an adjustable threshold between low and high w_{slow} values. In order to avoid resetting premature populations, for which w_{slow} had not enough time to exceed 1, this rule is applied only to converged populations.

Rule 3 The first two rules cover most of the cases when population reset is needed. However, it is not certain that each population eventually converges and also w_{slow} can be low for an extended period of time, so there is a need for one more rule to trigger population reset in such situations. It would simply reset any population which is old, but not converged. In this case, the probability of reset is defined as

$$pr_3 = \min(\frac{A}{\alpha_2}, 1) \quad ()$$

where A is the population age and $\alpha_2 \in N^+$ is the maximum age, both expressed as the number of simulation steps.

All three rules are defined in probabilistic terms, rather than in deterministic terms, in order to utilize complete range of values of w_{slow} and w_{fast}. We expect such formulation to result in a greater flexibility of our method and ability to work with a wider range of maps. Probabilistic approach is also beneficial from the practical point of view, because the total number of parameters needed to define all rules is reduced to only two α_1 and α_2).

As it can be seen in Algorithm 3, all three rules are independent since the random uniform variable $U(0,1)$ is drawn individually for each rule. Population reset rules are placed at the end of DEMCL algorithm (Algorithm 3).

Algorithm 3 population_reset (X_t, w_{avg}, A, α_1, α_2)

1: $w_{slow} = w'_{slow} + 0.2 \cdot (w_{avg} - w'_{slow})$
2: $w_{fast} = w'_{fast} + 0.95 \cdot (w_{avg} - w'_{fast})$

3: $p_{r1} = \min(0, 1 - \frac{w_{fast}}{w_{slow}})$
4: $p_{r2} = \min(0, \alpha_1 - w_{slow})$
5: $p_{r3} = \min(\frac{A}{\alpha_2}, 1)$

6: **if** $U(0,1) < p_{r1}$ **then**
7: Reset population X_t
8: **end if**
9: **if** X_t is converged AND $U(0,1) < p_{r2}$ **then**
10: Reset population X_t
11: **end if**
12: **if** X_t is not converged AND $U(0,1) < p_{r3}$ **then**
13: Reset population X_t
14: **end if**

Implementation of the resetting rules is relatively simple, since it requires only little additional effort to track population age A and defining convergence criteria (Algorithm 3). In our case, population is being considered as converged if all particles are within 0.25m from the average pose estimated by this population.

IV. EXPERIMANTAL RESULTS

In this section, DEMCL was compared with MCL in terms of robot pose tracking. We are interested in localization speed, pose tracking accuracy and computation time. The proposed modifications to DEMCL algorithm also was evaluated in terms of robot kidnap problem and global localization.

A. *Comparing DEMCL with MCL*

In terms of localization speed, experimental results for MCL and DEMCL are very similar. Figure 2 presents ratio of successful trials which were localized before or exactly at the given step. In other words, all successfully localized trials until given step number are summed up and divided by the total number of successful trials in the experiment.

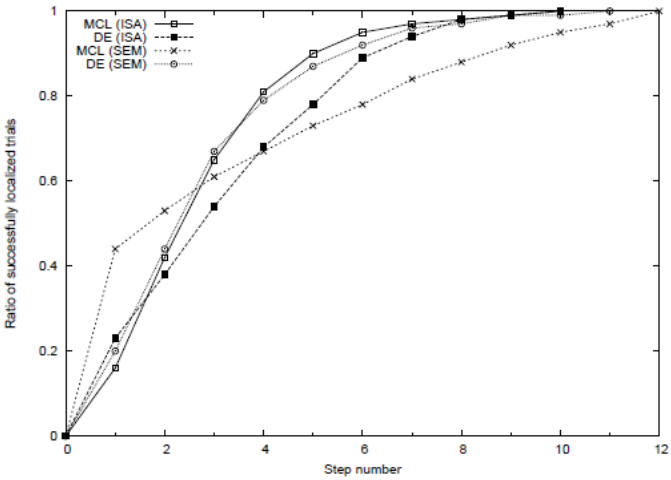

Figure 2. Ratio of successfully localized trials as a function of step number (time).

In terms of localization speed, results obtained show that the number of particles has little influence on pose tracking accuracy, therefore Table II presents only the average results.

On average, DEMCL method is 44% (ISA) and 50% (SEM) more accurate at pose tracking than MCL. Yet, both methods yield results in 0-0.1m range, which should be satisfactory for most real-life applications.

TABLE II. MCL AND DEMCL AVERAGE POSE TRACKING ACCURACY COMPARISON.

MCL	*DEMCL*
ISA 0.072±0.010m	0.050±0.016m
SEM 0.084±0.016m	0.056±0.009m

Increased computation time is the cost of significantly improved DEMCL effectiveness. Table III presents average computation time in ms for 100 particles. In this aspect, DEMCL performance is 53 (ISA map) and 17 (SEM map) times worse than MCL. Such a big difference is surprising, although an increase in computation time could not be avoided due to additional DEMCL calculations in each simulation step. On the other hand, DEMCL was using roughly 10 times less particles than MCL, which reduced the computation effort for DEMCL.

It is important to notice that exact computation time of DEMCL depends highly on the number of generations. Values presented in Table III were calculated using results from experiments with 30 (ISA map) and 10 (SEM map) generations. Any change in the number of generations would influence computation time of DEMCL, so it is not possible to directly compare MCL and DEMCL. Therefore results in Table III should be considered as a general indication of significant computation time increase, rather than as an exact difference.

TABLE III. COMPUTATION TIME IN MS PER 100 PARTICLES.

MCL	*DEMCL (ISA)*	*DEMCL (SEM)*
1.7	89.8	28

The most important observation is DEMCL ability to achieve much better success rates with fewer particles compared to MCL. DEMCL performance is significantly better than MCL in terms of effectiveness and pose tracking accuracy, however it requires much longer computation time.

B. Improved DEMCL

We expect resetting rule 1 to have the most critical influence on the results, therefore two variants of modified DEMCL are tested: with rule 1 only and with all three rules. In addition, global localization results are also presented for DEMCL without modifications. In case of robot kidnap problem, MCL results are included for reference purposes.

Global localization experiment consisted of 100 independent trials with exactly the same setup. At the beginning of each trial particles were initialized randomly and uniformly all over the free space indicated by the map. Next, at each simulation step (single iteration of DEMCL or MCL algorithm) position error was calculated as a distance between the estimated pose and the true robot pose. If this error was less than 0.25m for 5 consecutive steps, global localization was considered to be successful in the particular trial. The maximum length of each trial was set to 50 steps. Starting point in the map was selected randomly for each trial ensuring odometry and sensor data were available at least for 50 steps.

Results presented in Table IV clearly indicate that population resetting with rule 1 gives the best results, except for SEM map. In general, introducing any of the two resetting strategies improves global localization results. The explanation might be the fact that without resetting rules population has only one chance to localize the robot. With resetting there can be numerous attempts to localize the robot as a result of particles being again initialized randomly in the complete map.

TABLE IV. GLOBAL LOCALIZATION RESULTS SUMMARY FOR DEMCL VARIANTS. LGORITHMS ARE ORDERED STARTING FROM THE MOST SUCCESSFUL ONE.

Map	*DEMCL (ISA)*		
ISA	Rule 1	Rule 1-2-3	No resetting
SEM	Rule 1-2-3	Rule 1	No resetting
UWB	Rule 1	No resetting	Rule 1-2-3
COR	Rule 1	Rule 1-2-3	No resetting

Robot kidnap experiment also consisted of 100 independent trials, however the setup was different compared to global localization experiment. First of all, at the beginning of each trial particles were initialized only locally around the random starting position of the robot to avoid the need of global localization. Once the pose tracking was reliable, robot true pose was changed randomly. When pose error decreased to less than 0.25m, within 50 steps from the moment of kidnap, the robot was considered as successfully recovered from kidnap and the trial was counted as successful.

Values of w_{slow} and w_{fast} during a typical trial are presented in Figure 3. In addition, Figure 3 includes the average ratio of new particles created by DE at each generation, which we define as

$$\gamma_{new} = \frac{1}{g_{max}} \sum_{g=1}^{g_{max}} \frac{N_{new}^{g}}{P} \qquad ()$$

where $P = const$ is the total number of particles and N_{new}^{g} is the number of new particles created from population P_g which have higher fitness values than the parents and therefore are added to P_{g+1} (Algorithm 3, lines 13-17). Parameter γ_{new} indicates how effective is the evolutionary search in DE and also how likely it is for a particle to be replaced by a mutated candidate particle.

At the beginning of the trial (Figure 3, steps 1-10), all three parameters are very low. In case of w_{slow} and w_{fast} this is expected because robot is not yet localized, so particle weights are low. Surprisingly, γ_{new} is also very low (~6%), which might

indicate that evolutionary search in DE is not effective, when it is needed the most. However if we take into account the number of generations g_{max}=30, than we can easily calculate that each particle is on average replaced by a better candidate almost twice at each simulation step. This brings us to conclusion that even when γ_{new} is low the particle set is still evolving.

At step 11 w_{fast} rapidly increases which indicates that obtained measurements start to match the expected ones very well. High w_{fast} might signal successful global localization, even though it is not always a sufficient indicator. In general, there might be more than one area in the map resulting in high correlation with obtained measurements. At the same time, also γ_{new} increases from ~6% to ~30%, which suggests that particle set is evolving around five times more often than at the beginning of the trial. This observation might be explained by the fact that it is more likely for a candidate particle to have high fitness value, when particles are converged around the true robot pose, because then mutation acts only locally.

Next, w_{slow} is slowly rising, due to constantly high w_{fast}, and reaches maximum value at step 27. At this point in the simulation, both global localization and pose tracking prove to be successful, so robot kidnap occurs. As a result of sudden change in robot true pose, w_{fast} quickly decreases from 0.97 at step 27 to only 0.07 in the following step, because measurements are not aligned with expectations any more. However, w_{slow} is still quite high (change from 0.94 to 0.76) so the probability of population reset according to rule 1 is high and equal to $p_{r1} = 1 - w_{fast}/w_{slow} = 1 - 0.07/0.76 = 0.91$.

As a consequence of population reset at step 28, both w_{slow} and w_{fast} are set to 0. As seen in Figure 3, at step 35 w_{fast} again suddenly increases and w_{slow} also catches up within steps 35-50, which in this trial indicates successful robot pose recovery after kidnap.

To sum up, in this example trial robot kidnap was quickly detected after one steps (28-29) and after following six steps (29-35) robot pose estimate was successfully recovered.

Overall results from the complete experiment (Table V) prove that DEMCL with resetting rule 1 is clearly the most successful algorithm in solving the kidnap problem. Algorithms are ordered starting from the best.

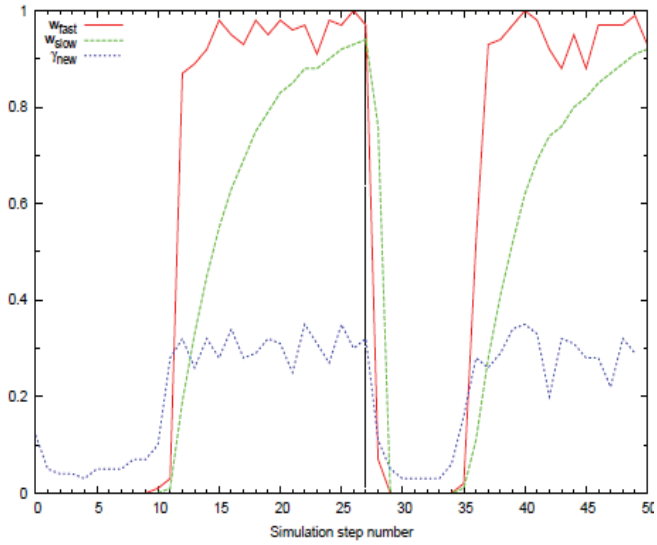

Figure 3. Population parameters in kidnap experiment: long-term (wslow) and short-term (wslow) average particle weights, average ratio of new particles (new) introduced by DE at each generation. Robot kidnap occured at step 27 (indicated by vertical line).

TABLE V. ROBOT KIDNAP OVERALL RESULTS FOR DEMCL AND MCL.

Map	***DEMCL (ISA)***		
ISA	DEMCL (rule 1)	DEMCL (rule 1-2-3)	MCL
SEM	DEMCL (rule 1)	DEMCL (rule 1-2-3)	MCL
UWB	DEMCL (rule 1)	DEMCL (rule 1-2-3)	MCL
COR	DEMCL (rule 1)	DEMCL (rule 1-2-3)	MCL

V. CONCLUSIONS

In this paper, both MCL and DE algorithms have been investigated and demonstrated how they can be easily combined to achieve increased performance. The outcome, modified DEMCL algorithm, is a complete solution for all the localization sub-problems. Experimental evaluation of the proposed algorithm proved significant improvements compared to MCL.

Regarding the challenging robot kidnap problem, the most important advancement was observed, due to the introduced population resetting rule. With sufficient number of particles, modified DEMCL algorithm was able to recover robot pose within 5 to 10 steps after kidnap with effectiveness reaching 100% on 3 out of 4 tested maps (except UWB). Whereas MCL results indicated constantly low effectiveness in recovery from kidnap, with success rates only up to 30% on most favorable map (SEM).

In case of global localization, proposed algorithm demonstrated ability to achieve the same success rates as MCL but with significantly fewer particles. The reduction ratio in particle set size ranges from 14 to 40 times. MCL solution to pose tracking was satisfactory with an average error of 7-8cm.

However modified DEMCL algorithm results in an average error of 5-6cm, which is slightly better.

The future work will focus on the investigation in the following areas: measurement model (fitness function), the improvement in computation cost and efficiency, multiple populations concept applied in DEMCL, and further DE analysis on other mutation and crossover variants or more advanced DE algorithms e.g. based on stochastic ranking. This issue could lead to further improvement of modified DEMCL algorithm and better performance results.

REFERENCES

[1] L. Moreno, S. Garrido, and ML Munoz. Evolutionary filter for robust mobile robot global localization. Robotics and Autonomous Systems, 54(7):590–600, 2006.

[2] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. Artificial Intelligence, 128(1): 99–141, 2001.

[3] Jose-Luis Blanco, Javier Gonzalez, and Juan-Antonio Fernandez-Madrigal. An optimal filtering algorithm for non-parametric observation models in robot localization. In IEEE International Conference on Robotics and Automation, 2008, pp. 461–466.

[4] L. Ronghua and H. Bingrong. Coevolution Based Adaptive Monte Carlo Localization (CEAMCL). International Journal of Advanced Robotic Systems, 1(3):183–190, 2004.

[5] A. Gasparri, S. Panzieri, F. Pascucci, and G. Ulivi. Monte carlo filter in mobile robotics localization: A clustered evolutionary point of view. Journal of Intelligent and Robotic Systems, 47(2):155–174, 2006.

[6] L. Moreno, J.M. Armingol, S. Garrido, A. De La Escalera, and M.A. Salichs. A genetic algorithm for mobile robot localization using ultrasonic sensors. Journal of Intelligent and Robotic Systems, 34(2):135–154, 2002.

[7] R. Storn and K. Price. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. INTERNATIONAL COMPUTER SCIENCE INSTITUTE-PUBLICATIONS-TR, 1995.

[8] G.C. Onwubolu and D. Davendra. Differential Evolution: A Handbook for Global Permutation-based Combinatorial Optimization. Springer Verlag,2009.

[9] T. Higuchi. Monte Carlo filter using the genetic algorithm operators. Journal of Statistical Computation and Simulation, 59(1):1–23, 1997.