Polibits

ISSN: 1870-9044

polibits@nlp.cic.ipn.mx

Instituto Politécnico Nacional

México

Moravec, Jaroslav

Map Building of Unknown Environment Using L1-norm, Point-to-Point Metric and Evolutionary Computation

Polibits, vol. 46, 2012, pp. 29-38

Instituto Politécnico Nacional

Distrito Federal, México

Map Building of Unknown Environment Using L1-norm, Point-to-Point Metric and Evolutionary Computation

Jaroslav Moravec

***Abstract*—In the present paper, a method for building a map of an unknown environment (SLAM) derived from the ICP algorithm using point-to-point metric is proposed. The polar-scan matching technology is used for estimation of the robot location change between two scans in sequence estimate the correct position of the robot. Since map building is fairly time-consuming, the algorithm of differential evolution (DE) is used in the calculation. This efficient optimizer provides very good results in different types of small office environment (unstructured and structured). The new type of an algorithm for map building is based purely on simple geometric primitives—vectors and integrates the modern evolutionary algorithm—DE. The presented algorithm falls into the wider group of geometric map builders and is able to build a map of indoor, mostly office, environment without moving objects.**

***Index Terms*—SLAM, robot localization, evolutionary robotics, differential evolution, L1-norm.**

I. Introduction

SIGNIFICANT effort of many different research groups in the area of the map building has brought good results in the last several decades. The integration of modern evolutionary algorithms is not taken for granted that much in this field. Disadvantage of nearly all EA (evolutionary algorithms) methods is a necessity to find proper working parameters. Many EA methods suffer from premature convergence to local optimum, which they´re not able to release from any more. Algorithms for the map building are very sensitive about the failure of the estimator which performs the estimation of position and turning transformation. All these exact reasons lead to elect differential evolution optimizer as an appropriate EA tool, as it provides very good results for a given task.

There are many different approaches in an area of the robot localization and map building which can be classified into several main groups. The amount of publications in particular groups is approximately the same. (A) Probabilistic algorithms usually use different versions of an occupancy grid. A map is represented by set of occupancy probability eventually emptiness probability. The map is formed by a set of cells in the shape of usual square area [23, 9, 3, 13]. (B) Map is represented by geometric primitives such as lines, circles, points or curves (b-spline curves) etc. The geometric model of environment is created from these elements. The core of the position estimator might be compiled by using various types of algebraic criteria [18, 19, 20, 21]. (C) The third and these days probably the widest group is the one using combinations of algorithms from two previous groups (so called hybrid algorithms) or it uses very specific patterns that represented the model of the world and different methods for localization and map building. It´s for example the case of events when a map is represented by a cloud of points, alternatively by different kinds of landmarks [17] like RFID, sound sources etc. Relatively new and perspective way in the field of map building and navigation is called cognitive maps [38, 39]. This approach exploits and integrates more information sources. A precise geometric map similar to [13] does not exist here.

Conventional methods for creating a map of unknown environment such as e.g. these publications [23, 1, 21] use gradient optimizers. But this is an approach a few decades old. The advantages of gradient optimizers are their simplicity and implementation speed. They still interest many researchers thanks to these qualities. They may be found e.g. in [40]. However, they have their insignificant limitations. Due to an intensive research in the field of evolutionary computer technology and fairly huge amount of publications analyzing their possibilities on different types of problems, EA methods have come to the foreground in the area of robotics as well. Their application is broad – map building using 2D or 3D laser scanner, global and local localization, semantic classification, the area of machine learning. A relatively big disadvantage is that they may also extend significantly the implementation of a basic navigation algorithm. MoteCarlo algorithm is the most common optimizer that is possible to come across and is used in the connection with probabilistic algorithms [9, 3, 13].

In 1998 an interesting article [42] based on Island Model Genetic Algorithm (IGA) was published. The theme of distributed GA can be found earlier for example [43, 44]. IGA is a derivative of the genetic algorithms that works with several populations which search functional space in parallel. The authors were successful to prove that IGA provides better results especially for linearly separable problems comparing to SGA [48] that is used e.g. in [26]. Using IGA optimizer as a computational accelerator also depends strongly on a type of the basic method(s). These methods were used for the purposes of localization and map creation (SLAM) in [45], [46]. It is also possible to find a very interesting connection of the SLAM algorithm and the fuzzy logic [47]. One of the first papers using untreated SGA was published in [11]. Interesting

Manuscript received June 20, 2012. Manuscript accepted for publication July 24, 2012.

J. Moravec is with the Czech Technical University in Prague, Prague, Czech Republic (e-mail: j.moravec.email@seznam.cz; webpage and source code: robomap.4fan.cz, www.openslam.org).

results like that may be found in the article [55] as well. Practical use of the SLAM algorithm with the complete analysis of the issue can be found e.g. in [25]. The base of the presented method is formed by the probabilistic occupancy grid [23, 13] again. Kwok et al. presented a small study in [16] which compares the performance of three different evolutionary algorithms SGA [48, 49], PSO [53] and AntSystem [52]. The tested EA methods solve three-dimensional problem – here the dimensions are represented by X, Y coordinates in the Cartesian coordinate system and the third dimension is then the angle α that includes the robot´s view direction along with axis X. The disadvantage of this approach is the necessity of using a fairly huge number of individuals in the population and substantially higher computation demands related to that. Similar results are to be found in paper [24] as well.

The differential evolution algorithm (DE) proposed by Kenneth Price a Rainer Storn [36] is used in this paper as an efficient and powerful computational accelerator (optimizer). However, DE is only suitable for certain types of problems – see [14]. SLAM under low noise levels falls to such suitable groups of problems as well. Finding of the correct working parameters of the optimizer is not so easy and usually takes a fairly long time. Valuable results of a research in this area can be found in [2, 12, 37, 14, 30, 22].

II. Pose Estimation in Partially Known Environment

Consider a general evolutionary SLAM problem

$$min_{x\in\boldsymbol{X}} f\big(g(x)\big) \text{ or } max_{x\in\boldsymbol{X}} f\big(g(x)\big) \tag{1}$$

Denote by f_{OPT+}, f_{OPT-} the optimal values of this problem and by f^* and f_* the maximum and minimum value of f over $\boldsymbol{X}$. $x \in \boldsymbol{X}$ represents optimal trajectory in state space (it is for example: x, y, α is a robot pose; x, y in Cartesian coordinates and α is heading with regard to the axis X, and of course all working parameters of the presented methods have to be included as well). $g(\dots)$ represents sensing model and a pose estimator (in the case of the SLAM problem, presented here, it is $\mathcal{F}_2$ or $\mathcal{F}_3$ strategy – see below), $f(\dots)$ represents evolutionary pseudo-random process – i.e. DE algorithm for example.

Definition: Given $\varepsilon \in \langle -\infty, +\infty \rangle$, a functional $\tilde{x} \in \boldsymbol{X}$ is said to be an ε-approximate solution of the problem (1) if possible solution exists in the sense

$$\begin{aligned} |f(\tilde{x}) - f_{OPT-}| \le \varepsilon |f_* - f_{OPT-}| \text{ or} \\ |f(\tilde{x}) - f_{OPT+}| \le \varepsilon |f^* - f_{OPT+}| \end{aligned} \tag{2}$$

Unfortunately, $\tilde{x} \in \boldsymbol{X}$ strongly depends on $g(\dots)$ and "system" represented by robot, environment and all moving objects and is non-separable and non-stationary (so called t-variant). In this task robot always affects itself through other objects moving in the given environment and that´s thanks to the used control systems. The presented work transforms the general optimization problem utilizing evolutionary computation to:

$$\begin{aligned} \mathcal{F}_{2,3}: f\big(g(x)\big) \rightarrow f_{EA}\big(g(x)\big)\,,\ \boldsymbol{D}(f, f_{EA}) \in \mathbb{R}, \\ f_{EA} \sim \textit{pseudo-random process} \end{aligned} \tag{3}$$

Evolutionary computations are used to accelerate the map building process. The pose estimation process is based on comparison of a set of simulated data from a virtual 2DLS (two dimensional laser scanner) from positions obtained by using the EA process and scan from the real 2DLS. We can denote: $P_{e(i)} = (x, y, \alpha)$, $P_{s(j)} = (x, y, \alpha)$, $i, j \in \mathbb{N}$, $\exists j | P_{opt} \equiv P_{s(j)} \sim P_{e(i)}$ is a real and simulated pose and from the text above $P_{e(i)}, P_{s(j)} \in \boldsymbol{X}$, $D_{e(i)} = \big[v_{e(1)}, v_{e(2)}, \dots, v_{e(m)}\big]$, $i \in \mathbb{N}$, $\forall v_e | v_e \le L_{max}$, $1 \le m \le \mathbb{C}_{real}$ and $D_{s(j)} = \big[v_{s(1)}, v_{s(2)}, \dots, v_{s(n)}\big]$, $j \in \mathbb{N}$, $\forall v_s | v_s \le L_{max}$, $1 \le n \le \mathbb{C}_{sim}$, $m < n$, the real and simulated sensing. $\mathbb{C}_{sim} \ge \mathbb{C}_{real}$. α is heading with regard to axis X. For every $P_{e(i)}$ it is necessary to create set of $P_{s(j)}, j \in \mathbb{N}$. $\varphi_{det} = \pi$ is a detection angle of the real 2DLS, $\varphi_{stp} = 0.5°$ is a resolution of the real 2D LS - i.e. 361 beams can be used for example. γ_{det} is a detection angle of the pose estimator $g(x)$ i.e. γ_{det} is the angle which the matching process works in. γ_{det} is set to $\pi/3$ for all experiments. $P_{opt}|_1^{\boldsymbol{n}} \in \tilde{x}$, L_{max} is a beam limitation given by used sensor with $\mathbb{C}_{real}$ beams. $\mathbb{C}_{sim}$ denotes the number of simulated beams. Environment is represented by set of short lines. EA methods use set of individuals $P_{s(j)}$. From the theoretical point of view, only one point $P_{opt}, P_{opt} \in P_s$ defines the correct robot's pose. In the real world and thanks to the existence of noise (estimated by S_{p3} or S_{p2} function), more than one point can provide well acceptable result(s). Every individual of the EA represents one possible solution which is evaluated in the sense:

$$\boldsymbol{C} = |\boldsymbol{A} - \boldsymbol{B}|, \tag{4}$$

where

$$\boldsymbol{A} = \begin{bmatrix} v_{s\prime(1)} & v_{s\prime(2)} & v_{s\prime(3)} & \cdot \\ v_{s\prime(2)} & v_{s\prime(3)} & \cdot & \cdot \\ v_{s\prime(3)} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ v_{s\prime(n-m+1)} & v_{s\prime(n-m+2)} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ v_{s\prime(n-3)} & \cdot & \cdot & \cdot \\ v_{s\prime(n-2)} & v_{s\prime(n-1)} & \cdot & \cdot \\ v_{s\prime(n-1)} & v_{s\prime(n)} & v_{s\prime(n+1)} & \cdot \\ v_{s\prime(n)} & v_{s\prime(n+1)} & v_{s\prime(n+2)} & \cdot \end{bmatrix}$$

$$\begin{bmatrix} \cdot & \cdot & \cdot & v_{s\prime(m-1)} & v_{s\prime(m)} \\ \cdot & \cdot & v_{s\prime(m-1)} & v_{s\prime(m)} & v_{s\prime(m+1)} \\ \cdot & v_{s\prime(m-1)} & v_{s\prime(m)} & v_{s\prime(m+1)} & v_{s\prime(m+2)} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & v_{s\prime(n-1)} & v_{s\prime(n)} \\ \cdot & \cdot & v_{s\prime(n-1)} & v_{s\prime(n)} & v_{s\prime(n+1)} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & v_{s\prime(n+m-3)} \\ \cdot & \cdot & \cdot & v_{s\prime(n+m-3)} & v_{s\prime(n+m-2)} \\ \cdot & \cdot & v_{s\prime(n+m-3)} & v_{s\prime(n+m-2)} & v_{s\prime(n+m-1)} \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} v_{e(1)} & v_{e(2)} & \cdot & v_{e(m)} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ v_{e(1)} & v_{e(2)} & \cdot & v_{e(m)} \end{bmatrix}, \boldsymbol{A}, \boldsymbol{B} \text{ are } n \times m.$$

$v_{s\prime(1..n)} = v_{s(1..n)}$, $v_{s\prime((n+1)..(n+m-1))} = v_{s(1..(m-1))}$;
$a \equiv v_{s\prime}$, $b \equiv v_e$; n, i is row, m, j is $column$;
$\boldsymbol{u^T} = \big[\sum_{j=1}^{m} c_{1,j}, \sum_{j=1}^{m} c_{2,j}, \dots, \sum_{j=1}^{m} c_{n-1,j}, \sum_{j=1}^{m} c_{n,j}\big]$

Matrices $\boldsymbol{A}, \boldsymbol{B}$ are expressed generally in (4) i.e. $\gamma_{det} = 2\pi$. If γ_{det} is smaller than 2π, number of rows of the used matrices will be adequately smaller too. Presented map building algorithm is based on two independent strategies. Both strategies $\mathcal{F}_2$ and $\mathcal{F}_3$ enable correct $\tilde{x} \in \boldsymbol{X}$ estimation. $\mathcal{F}_2$ and $\mathcal{F}_3$ are given by equations:

$$\mathcal{F}_2|_{P_{e(i=1)}}^{i=t} \equiv \left\{ S_{r2} = argmax_{P_s(x,y)} max_{u_{(i)}} |_{i=1}^{n} \left[\sum_{j=1}^{m} \begin{cases} 1, \ if \ (|a_{(i,j)} - \ b_{(i,j)}| \leq S_{p2}) \\ 0, \ if \ (|a_{(i,j)} - \ b_{(i,j)}| > S_{p2}) \end{cases} \right] \right\} \quad (5)$$

$$\forall x, y | x, y \in \mathbb{R} \ , S_{r2} \leftrightarrow P_{opt}; S_{p2} = \frac{1}{n} + \varepsilon_t (b_{i,j} + a_{i,j}) + 1, \varepsilon_t = \frac{59}{6000}$$

$$\mathcal{F}_3|_{P_{e(i=1)}}^{i=t} \equiv \left\{ S_{r3} = argmin_{P_s(x,y)} min_{u_{(i)}} |_{i=1}^{n} \left(\sum_{j=1}^{m} \begin{cases} |a_{(i,j)} - \ b_{(i,j)}|, if \ a_{(i,j)} < \ L_{max} \\ 0, \quad if \ a_{(i,j)} = \ L_{max} \end{cases} \right) \right\} \quad (6)$$

$$\forall x, y | x, y \in \mathbb{R} \ , S_{r3} \leftrightarrow P_{opt} \ , u_{(i)}|_{i=1}^{n} \in \{0,1\}$$

$$u_{(i)}|_{i=1}^{n} = \sum_{j=1}^{m} |a_{(i,j)} - \ b_{(i,j)}| \ ; S_{p3} = \left[\frac{1}{n} + (\varepsilon_t a_{i,j}) + 1 \right]$$

where ε_t is a slope of the accuracy curve of the used 2DLS (for example $\frac{59}{6000}$) – it is classic linear dependency according to the manufacturer recommendations. t is the number of all collected scans from the real 2DLS.

Here, S_{r2} and S_{r3} represents the fitness value of the best founded estimated pose. S_{p3} and S_{p2} represent equations of the linearized model of the 2DLS sensor—simple noise model for one(every) beam. Correct pose and heading estimation according to the selected strategy is given by P_{opt} and heading α of the robot is given by:

$$\boldsymbol{\mathcal{F}_3}: \ min \, u_k|_{k=1}^{n} \Rightarrow (index)k \ , \ \boldsymbol{\mathcal{F}_2}: \ max \, u_k|_{k=1}^{n} \Rightarrow (index)k \ ; \quad (6)$$

$$\boldsymbol{\mathcal{F}_{2,3}}: \alpha = \left(\varphi_{stp}(k-1) \right) + \left(\frac{1}{2} \gamma_{det} \right)$$

'$index$' means k-th element of the $u_k$vector, for which the fitness function takes the smallest ($\mathcal{F}_3$) or the biggest ($\mathcal{F}_2$) value.

Sensorial data from 2DLS are used only (no data from odometry). The pose estimator described in here is based on point-to-point metric. It is the core of the proposed SLAM method. $\mathcal{F}_2$ or $\mathcal{F}_3$ strategy is used to dissimilarity measurement—dissimilarity between simulated vector D_s and the real sensing D_e. $\mathcal{F}_3$ strategy has universal features and is suitable for structured or unstructured environment, long or small corridors (hallways) or environment with or without moving objects.

Generally, $\mathcal{F}_3$ provides somewhat worse results at heading estimation—of about 5–7 percent in comparison to $\mathcal{F}_2$. $\mathcal{F}_2$ has identical features to $\mathcal{F}_3$, but it is not suitable for work in long hallways and has quarter noise resistance abilities only – see [54]. $\mathcal{F}_2$ is suitable for small and very structured environment with or without moving objects.

At correct pose estimation and if searching area is 60x60cm for example, equation (4) must be evaluated 60 × 60=3600 times. It takes a long time. DE optimizer is able to estimate correct solution approx. 25–35x faster. Fig. 1 depicts Fitness function projection of the $\mathcal{F}_3$ and $\mathcal{F}_2$ strategy to 3D space—identical environment and identical robot's pose is used.

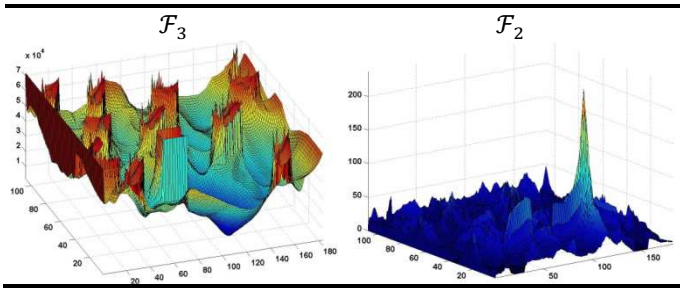


Fig. 1. Projection of the Fitness function of the $\mathcal{F}_3$ and $\mathcal{F}_2$ strategy to 3D space - example. $\mathcal{F}_3$ - robot is in the place with min. Fitness function. $\mathcal{F}_2$ - robot is in the place with max. Fitness function.

TABLE I
PROPOSED GSLAM ALGORITHM

Input data: $P_{actual}(x, y, \alpha)$ – Start (alias actual) robot's pose – corresponding to the $D_{e(1)}$, $D_{e(i)}$ – Set of sensorial data from 2DLS, $\boldsymbol{GM}$ - empty global map.		
A1	1	**Approximate scan $D_{e(1)}$ by set of lines $U_k, k \in \mathbb{N}, k \neq 0$**
	2	**Compute parameters of the U_k, necessary for SLAM.**
A2	3	**Insert the $U_k, k \in \mathbb{N}, k \neq 0$ into GM**
	4	**for all $D_{e(i)}$; $i \in \mathbb{N}, i \in \{2,..,p\}$**
A3	5	**Create TM based on the $P_{actual}(x, y, \alpha)$ _and GM.**
A4	6	**Find correct pose and heading P_{opt} for $D_{e(i)}$, _use the TM and selected strategy $\mathcal{F}_3$ or _$\mathcal{F}_2$ accelerated by DE.**
A1	7	**Approx. actual $D_{e(i)}$ vector, by set of _lines $U_k, k \in \mathbb{N}, k \neq 0$.**
	8	**Compute params. of the U_k and insert _U_k into the LM.**
A2	9	**Use the LM, build a new set of lines U_{new}, _suitabe to be inserted into GM. _Clear LM. Insert the new set of _lines U_{new} into LM.**
	10	**Insert all lines from LM to GM and _clear LM.**
A5	11	**Use heuristic rules, merge all possible _lines in GM, if it is possiblle.**
	12	**end for**
A5	13	**Final map 'refinement' – GM map**
Output data: $\boldsymbol{GM}$ – Lines list – global map of environment, $P_k(x, y, \alpha)$ – Set of positions. Robot's pose and heading corresponding with $D_{e(i)}$ vectors.		
Key: $\boldsymbol{TM}$ – Temporary Map of local environment. This map is used at pose estimation utilizing EA computations. $\boldsymbol{LM}$ – Local Map contains all lines found in $D_{e(i)}$. A1.. A5 individual parts of the gSLAM algorithm. A1 consists of a recursive line splitting marked as B1 and line pose improvements by LSQ algorithm marked as B2. A3 use classic ray-tracing. In step A4, EA method is used according to the equation (4). Line 13 can be optionally omitted.		

DE optimizer in this case seeks for an optimum (minimum or maximum) according to the u_k (alias $\mathcal{F}_3$: $min \, u_k|_{k=1}^{n}$, $\boldsymbol{\mathcal{F}_2}$: $max \, u_k|_{k=1}^{n}$, see (3) – (6)) value for every individuum of the population. Fig. 1 shows large area 600 × 800cm, for better understanding.

III. GSLAM ALGORITHM

Proposed gSLAM algorithm uses raw 2DLS data to estimate the correct pose in polygonal environment by modified simulated-point-to-point matching technique in

temporary map $\boldsymbol{TM}$ which represents the temporary polygonal model-view from the last estimated correct pose P_{opt} in $\boldsymbol{GM}$ – global map. $\boldsymbol{TM}$ is obtained by using 2D ray-tracing. The complete gSLAM algorithm is described in Table I. Once the correct robot's pose is found according to the $\boldsymbol{GM}$, the actual sensorial data $D_{e(i)}$ are approximated by set of lines and the local model $\boldsymbol{LM}$ is created from them. The set of lines is confronted with the global $\boldsymbol{GM}$ map of environment (previously built) and some parts of lines in $\boldsymbol{LM}$ are marked as suitable for inserting into $\boldsymbol{GM}$. This process uses several heuristic rules.

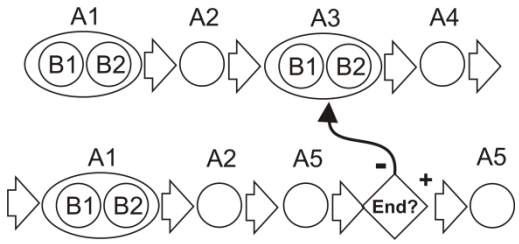
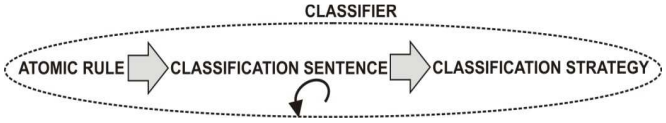


Fig. 2. Flowchart diagram of the gSLAM – sequential ordering of the A1 .. A5 algorithms.

The flowchart diagram of insertion process is depicted in Fig. 3. The recursive line splitting algorithm (LR-LSQ) based on L1-norm is used. 'LocalMap($\boldsymbol{LM}$)' serves like a helper only. Table I contains individual steps of the proposed gSLAM algorithm. gSLAM consists of 5 main parts. Input data are $P_{actual}(x, y, \alpha)$ – start (alias actual) robot's pose – corresponding to the $D_{e(1)}$, $D_{e(i)}$ – set of sensorial data 2DLS. Output data are $\boldsymbol{GM}$ – global map of environment, $P_k(x, y, \alpha)$ – set of estimated positions resp. robot's pose and heading corresponding with $D_{e(i)}$ vectors.



* A small circular arrow means that a classification sentence can be repeated according to the actual state of the $\boldsymbol{LM}$ map, until all abscissae are successfully classified. Deadlock is handled.

Fig. 3. The flowchart diagram of the lines relationship classifier.

A. The Line Fitting Method

The line fitting algorithm (marked as A1 in Fig. 2) uses approximation of a point set by multi-line. The presented line fitting algorithm uses combination of several methods - successive Edge Following – SEF [31] and Iterative End Point Fit (IEPF) [10, 6, 27, 28, 29]:

1) Transform D_e vector from polar to Cartesian coordinates.
2) Eliminate all points b_i where: $\forall b_i(x, y)$, $\nexists b_j(x, y)$ so that $b_i(x, y) \in \{b_j(x, y): \|b_i, b_j\| \leq \delta, j \neq i\}$ in E^2, δ is constant; 10cm for example.
3) Use SEF algorithm to b_i points and exclude all unsuitable points from b_i. New set of points will be b_i^*. Use IEPF algorithm to b_i^*.
4) Use linear regression (LR-LSQ) [32], [15] algorithm to merge two consecutive lines if it is possible + final refinement of every line in the sense of (LR-LSQ) algorithm.

Abscissae of the $\boldsymbol{LM}$ map cannot be inserted directly. Several atomic rules were defined enabling to estimate correctly, which part of the line is to be inserted into the $\boldsymbol{GM}$ – see Table II. These rules are designed to be useful for line-to-line algorithms. There are 17 atomic rules – see Fig. 4. Suitable combinations of atomic rules form four classification sentences. The classification sentences form a classification strategy. The number of applied classification sentences is not known a priori and is variable. The proposed classification sentences (insertion process of the $\boldsymbol{LM}$ into $\boldsymbol{GM}$):

1) If two abscissae AB and CD do not lie on an identical line (consecutively), the distance of C and D points from the line identical to abscissa AB is shorter than the limit, CD abscissa does not overhang the boundary C and D points of AB abscissa and the angle between AB and CD is smaller than the limit, remove CD abscissa. (Atomic rules 1,11, 12, 5, 4)
2) If two abscissae AB a CD do not lie on an identical line (consecutively), C point and D point overhang the end points of AB abscissa, the distance of A and B points to CD abscissa is shorter than the limit and the angle between AB and CD is shorter than the limit, let in $\boldsymbol{LM}$ parts of CD abscissa, which overhang AB abscissa. (Atomic rules 1, 3, 4, 7, 8, 9, 10, 13, 14, 16).
3) If two abscissae AB a CD do not lie on an identical line (consecutively), C point overhangs AB abscissa and the distance of the second point to AB abscissa is less than the limit, let in $\boldsymbol{LM}$ part of CD abscissa, which overhangs AB abscissa only. (Atomic rules 1, 2, 3, 7, 12, 13, 14, 15).
4) If two abscissae AB a CD do not lie on an identical line (consecutively), D point overhangs AB abscissa and the distance of the second point to AB abscissa is less than the limit, let in $\boldsymbol{LM}$ part of CD abscissa which overhangs AB abscissa only. (Atomic rules 1, 2, 4, 6, 12, 13, 16, 17).

The atomic rules are formularized by AND logic operator and rules 16 and 17 are placed inside of the 4. condition clause.

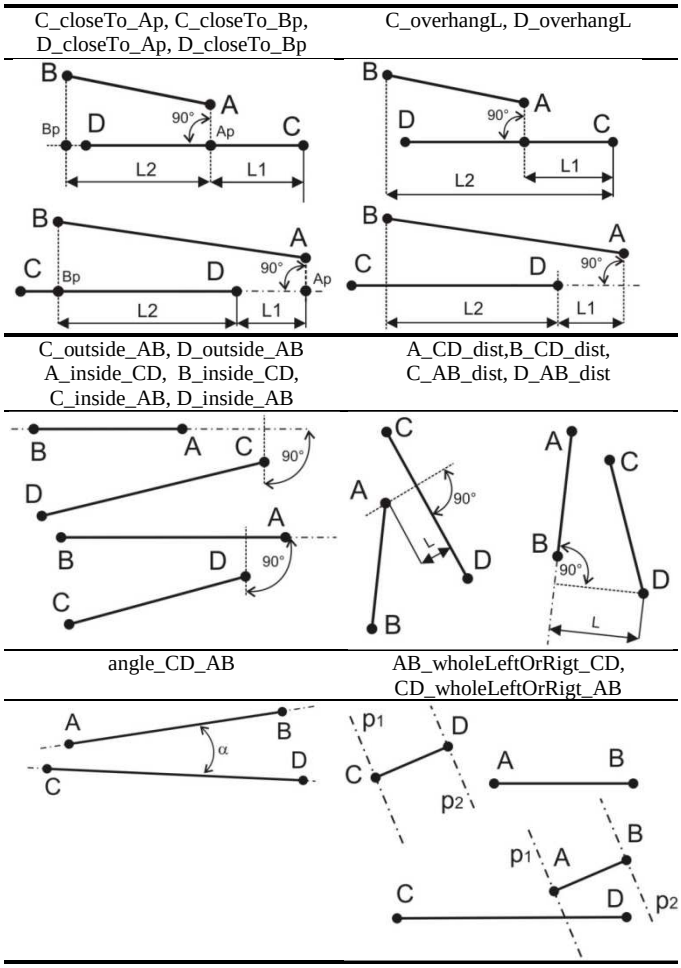
Four classification sentences represent a heuristic schema that only enables to define such parts of the $\boldsymbol{LM}$ map which are suitable for inserting into the $\boldsymbol{GM}$ map. If any "less suitable" abscissae appear in $\boldsymbol{LM}$, classifier inserts such abscissae into $\boldsymbol{GM}$ without any change. Usually perpendicular abscissae are considered – perpendicular to existing walls of the environment model.

TABLE II
ATOMIC RULE LIST – INSERTION PROCESS

Nr.	Atomic rule	Description
1	`AB_wholeLeftOrRigt_CD`	Abscissa AB lies on the left or right side of CD abscissa.
2	`CD_wholeLeftOrRigt_AB`	Abscissa CD lies on the left or right side of AB abscissa.
3	`C_overhangL <= L1`	Point C of CD abscissa does not overhang any of A or B points
4	`D_overhangL <= L1`	Point D of CD abscissa does not overhang any of A or B points
5	`C_overhangL > L1`	Point C of CD abscissa overhangs one of the outer points of AB abscissa AB.
6	`D_overhangL > L1`	Point D of CD abscissa overhangs one of the outer points of AB abscissa.
7	`C_outside_AB = true`	Intersect Cp of the line passing the point C perpendicular to AB abscissa does not lie between AB points.
8	`D_outside_AB = true`	Intersect Dp of the line passing the point D perpendicular to AB abscissa does not lie between AB points.
9	`A_CD_dist < L2`	Distance of the point A from the line which the CD abscissa lies on.
10	`B_CD_dist < L2`	Distance of the point B from the line which the CD abscissa lies on.
11	`C_AB_dist < L2`	Distance of the point C from the line which AB abscissa lies on.
12	`D_AB_dist < L2`	Distance of the point D from the line which the AB abscissa lies on.
13	`angle_CD_AB < L3`	Angle of AB and CD lines
14	`C_closeTo_Ap = true`	Distance of the C point and intersect Ap line passing the point A perpendicular to line on which the points C,D lie is shorter than distance of the point C and intersect Bp line passing the point B perpendicular to CD.
15	`C_closeTo_Bp = true`	Distance of the point C and intersect Bp line passing the point B perpendicular to line on which the points C,D lie is shorter than distance of the point C and intersect Ap line passing the point A perpendicular to CD.
16	`D_closeTo_Ap = true`	Distance of the D point and intersect Ap line passing the point A perpendicular to line on which the points C,D lie is shorter than distance of the point D and intersect Bp line passing the point B perpendicular to CD.
17	`D_closeTo_Bp = true`	Distance of the D point and intersect Bp line passing the point B perpendicular to line on which the points C,D lie is shorter than distance of the point D and intersect Ap line passing the point A perpendicular to CD.

If the rule has an operator (<,>,= et.c) it is only used in this form. If no operator is present, any type of operator can be used in algorithm in the sense of the particular rule. For example '`C_overhangL`' is real a number at computations and '`L1`' is constant.

If any abscissa is transformed using any classification sentence, only suitable parts of it are moved back to ***LM*** map. Once classifier finishes its job, all abscissae are moved to ***GM*** at once. Fig. 4 shows a graphical representation of the atomic rule list in Table II. used at an insertion process. It is a classic conceptional relation between two abscissae AB and CD.



Examples of the presented rules:
- C_closeTo_Ap = true, C_closeTo_Bp = false,
- D_closeTo_Ap = true, D_closeTo_Bp = false
- C_overhangL = L1, D_overhangL = -1*L1
- C_outside_AB = true, D_outside_AB = false,
- C_inside_AB = false, D_inside_AB = true
- A_CD_dist = L, D_AB_dist = L
- angle_CD_AB = α
- AB_wholeLeftOrRigt_CD = true, CD_wholeLeftOrRigt_AB = false

*L, L1, L2, α are elected constants at SLAM process.

Fig. 4. Atomic rule list, conceptional relation between two abscissae AB and CD; graphic representation.

*B. Merging – minimizing the number of abscissae in **GM***

Once the list of abscissae suitable for inserting into ***GM*** is completed, it´s inserted into ***GM*** immediately. Merging process ensures minimum and acceptable number of abscissae in ***GM***. Merging process is not a necessary step in order the final map to be fully consistent. The method presented in here is based on Skrzypczyński [33], [34] and Crowley [6], [7], [8], but the method is significantly modified. Similarly to the insertion process, the merging process uses the same basic scheme as the insertion process – see Fig. 3. The proposed merging process consists of 3 classification sentences:

1) Abscissae AB and CD lie consecutivey in a line and the distance of end-points is shorter than the limit. AB and CD will be concatenaded. (Atomic rules 13, 11, 12, 1, 18, 19, 20, 21, 22, 23).

2) If abscissae AB and CD intersect and the distance of any point A, point B, point C or point D from intersect AB and CD is less than the limit, cut-off this small protrusion. (Atomic rules 24, 25, 26, 27, 28).

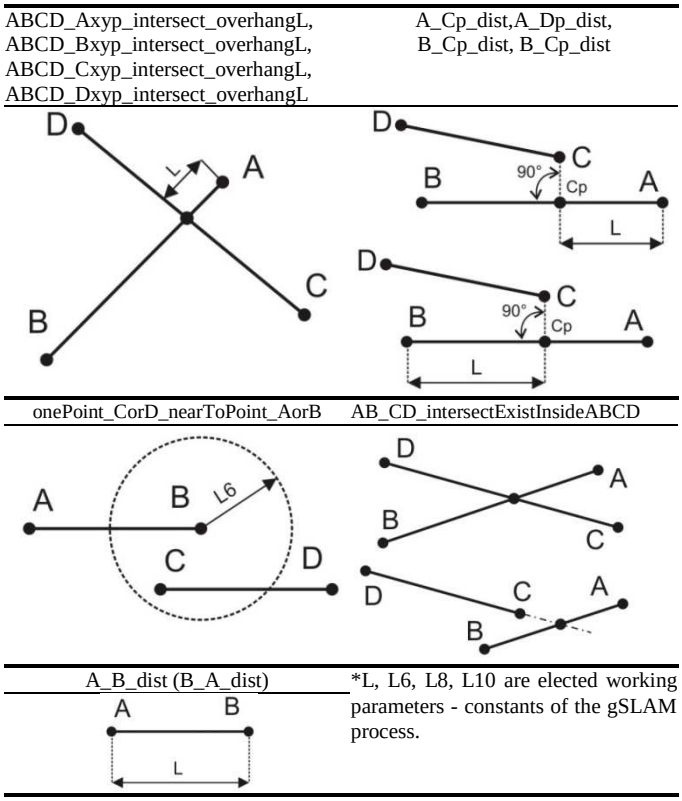
3) The intersect of the AB and CD does not esist and any point $p_1 = \{A, B, C, D\}$ is close to any other point $p_2 = \{A, B, C, D\}$, $p_1 \neq p_2$, and point p_1 or p_2 lies between AB or CD, merges with the nearest points. Only two points can be merged. (Atomic rules 24, 31, 35, 32, 36, 29, 33, 30, 33, 34).

TABLE III
ATOMIC RULE LIST – MERGING PROCESS

Nr.	Atomic rule	Description
18	onePoint_CorD_nearToPoint_AorB	One point C or D lie close to point A or B of AB abscissa. Max. distance is defined by fix. threshold.
19	A_B_dist, B_A_dist	Length of AB abscissa.
20	A_Cp_dist	Distance of the point A and intersect of AB and perpendicular line passing the point C.
21	A_Dp_dist	Distance of the point A and intersect of the AB and perpendicular line passing the point D.
22	B_Cp_dist	Distance of the point B and intersect of the AB and perpendicular line passing the point C.
23	B_Dp_dist	Distance of the point B and intersect of the AB and perpendicular line passing the point D.
24	AB_CD_intersectExistInsideABCD	This rule tells us that abscissae AB and CD have one intersect between AB and CD points. 'True' if yes, 'False' if intersect does not esist.
25	ABCD_Axyp_intersect_overhangL	Distance of the point A and intersect AB and CD abscissae, if rule 24 is true.
26	ABCD_Bxyp_intersect_overhangL	Distance of the point B and intersect AB and CD abscissae, if rule 24 is true.
27	ABCD_Cxyp_intersect_overhangL	Distance of the point C and intersect AB and CD abscissae, if rule 24 is true.
28	ABCD_Dxyp_intersect_overhangL	Distance of the point D and intersect AB and CD abscissae, if rule 24 is true.
29	A_inside_CD	Intersect Ap of the line passing the point A, perpendicular to CD is or is not inside CD abscissa. Rule can be true or false.
30	B_inside_CD	Intersect Bp of the line passing the point B perpendicular to CD is or is not inside CD abscissa. Rule can be true or false.
31	C_inside_AB	Intersect Cp of the line passing the point C perpendicular to AB is or is not inside AB abscissa. Rule can be true or false.
32	D_inside_AB	Intersect Dp of the line passing the point D perpendicular to AB is or is not inside AB abscissa. Rule can be true or false.
33	A_CD_dist	Distance of the point A from line which CD abscissa lies on.
34	B_CD_dist	Distance of the point B from line which CD abscissa lies on.
35	C_AB_dist	Distance of the point C from line which AB abscissa lies on.
36	D_AB_dist	Distance of the point D from line which AB abscissa lies on.

The merging process uses the atomic rule list mentioned in Table III – rules 18-36. Graphical representation of the presented atomic rules is depicted in Fig. 5.

The merging process occurs only in a limited range of possible combinations of AB and CD positions, which is also sufficient for the construction of a high-quality vector map. If unclassifiable schema appears, it is inserted into ***GM*** without any change. Such problems appear if random perpendicular abscissae, perpendicular to walls in environment, have to be processed for example.



Examples of the presented rules (from top to bottom, from left to right):
- (AB_CD_Axyp_intersect_overhangL > L8 & AB_CD_Axyp_intersect_ overhangL < L10)
- A_Cp_dist < A_Dp_dist, B_Cp_dist < B_Dp_dist
- onePoint_CorD_nearToPoint_AorB < L6
- AB_CD_intersectExistInsideABCD = true, AB_CD_intersectExistInsideABCD = false
- A_B_dist (B_A_dist) = L

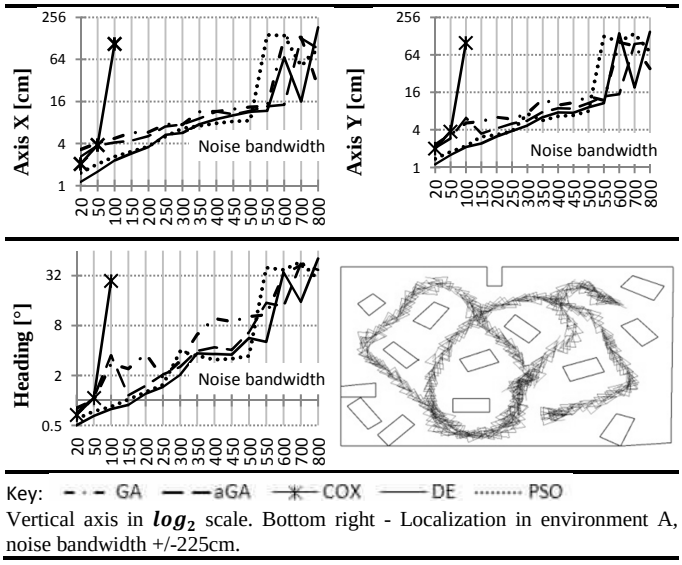
Fig. 5. Atomic rule list, conceptional relation between two abscissae AB and CD; graphic representation.

IV. EXPERIMENTAL RESULTS

A. DE efficiency and relevancy – short discussion

DE is a stochastic optimizer. The optimized task is continual, separable, t-variant, unimodal (for small searching area only). Several different evolutionary optimizers (EA) were tested in a continual localization task – see Fig. 6. All tested algorithms: SGA[48], [49], [55], aGA[54], PSO[53] and DE applied to (3) and (4) equations (alias $\mathcal{F}_2$, $\mathcal{F}_3$ strategy) provide well usable results. Beside these optimizers classic Cox's [5] gradient method was tested as well. All methods were tested under heavy-duty operation conditions at a continual localization task in known environment (known geometric map) to get their reliability and capabilities. Unfortunately, the map building algorithm is not highly noise resistant that much. Additive Gaussian noise with different

noise bandwidth (independent, fixed noise bandwidth) from zero to 800cm was tested. The sample of noise was obtained according to the equation: $D_{noisy\_e(i)} = D_{e(i)} + rnd(-L, +L)$, where $2L$ is equal to noise bandwidth and rnd represents a random numbers generator – normal distribution, mean zero. DE and PSO algorithms proved the best results without regard to different types of environments – see [54]. In contrast to other used EA, DE and PSO provide stable and almost identical results, especially on lower noise levels. Occasional malfunction (which SGA suffers from so much) was never observed. Permanent malfunction of any EA method caused by additive noise was observed only on highest noise levels from the level of about 550cm.



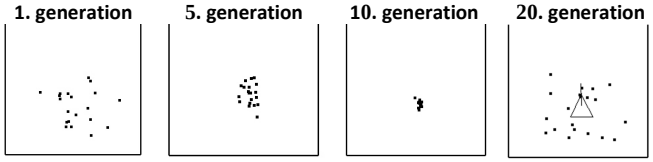
Vertical axis in $\boldsymbol{log_2}$ scale. Bottom right - Localization in environment A, noise bandwidth +/-225cm.

Fig. 6. Graphs of accuracy of estimating the position and heading - SGA, aGA, PSO, DE and Cox's optimizers.

For comparison classic, Cox gradient method is only able to work at noise level no higher than about 50. On Table 1, line 8, block A1, DE algorithm is used. Input of this optimizer is the area of width x height approx. 60 × 60cm (or bigger) around the last known robot's pose, $\boldsymbol{TM}$ (temporary map) obtained from $\boldsymbol{GM}$ based on classic ray-tracing algorithm + line fitting method – block A1; ($\boldsymbol{TM}$ is used for computation acceleration purposes only) and actual D_e vector.

DE optimizer used in presented gSLAM method solves a classic two-dimensional optimization problem. Heading of the robot is calculated separately because of the accuracy and higher speed. Step in heading is 0.5°. If a three dimensional optimizer would be used similar to [16], the number of generations and number of individuals in every population must be minimally three times higher.

In Fig. 7, population convergence is depicted. Individuals in the first generation cover equally the whole searching area (100x100cm is elected in here). After 10 generations, all individuals are almost at the correct pose. Normally 15 generations/10 individuals are efficient to ensure the correct convergence. After 20 generations, DE optimizer found the correct pose. In Fig. 7, robot is depicted by a small triangle; heading is depicted by a short line. 10 classic basic perturbation vectors were tested to get which vector is the most suitable. DE/Rand1/Exp provides the best results. Beside the DE, PSO optimizer provides good results too, but DE is better of approx. 5% if the additive (Gaussian) noise is small or zero. In structured environment PSO is significantly better from noise level approx. 250cm. Presented gSLAM algorithm is designed to build a map of unknown environment without moving objects and under noise stress no more than $\pm(8-10)$cm. Noise level value was obtained based on practical experiments. Ability to work under higher noise levels is significantly reduced thanks to A1 and A5 algorithms - line fitting and merging methods.



Key: 20 generations, 20 individuals, searching area 100x100cm. DE/Rand1/Exp, F=0.6,Pcr=1.0.

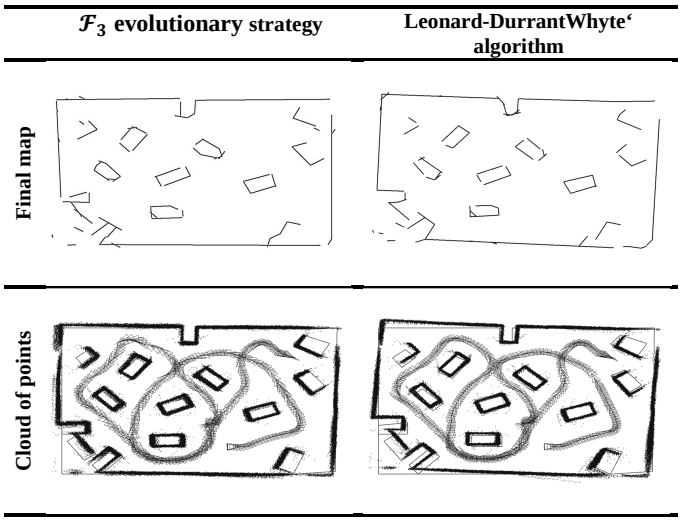
Fig. 7. Population convergence - Differential evolution.

B. Experimental verification of the gSLAM algorithm

The experimental verification of the proposed gSLAM algorithm was performed in two structured environments – A and B. The first environment was build-up for test purposes by cardboard boxes in laboratory. It is a common indoor office environment of 10 × 10 meters with several obstacles inside. The obstacles are cardboard boxes (approx. 40 × 60cm). Trajectory length was approx. 3001.11cm. The robot obtained 350 D_e vectors. The environment consists of 60 walls. Differential evolution - working parameters: DE/rand/1/exp, $POP = 25$, $GEN = 25$, searching area 32×32cm, $F = 0.6$, $P_{CR} = 0.9$. Such working parameters were obtained from practical experiments. If F value is smaller, time to correct pose evaluation is significantly longer. There is a linear dependence. Thanks to quantizing noise of used 2DLS Sick-PLS100 theoretical accuracy is ± 5 cm. Practically, it is twice as worse. The second environment B is a large cluster of offices. Dimensions are 2560 × 1880cm. Trajectory length is 23629.60cm. The number of lines is 329; the number of D_e vectors is 1832. Robot passed the trajectory which many times intersected itself. Environment B consists of 10 small offices and one long hallway. The working parameters of DE estimator were identical to the first experiment.

In the first experiment (see Fig. 8) robot passed the trajectory which intersects itself in three points. Presented estimator $\mathcal{F}_3$(or $\mathcal{F}_2$) does not use 'closing loop' mechanism (global localization based pose corrector) capable of improving the correct pose estimation globally. This makes the estimator more sensitive to noise.

The robot was able to pass through the environment without loss of orientation. Leonard-DurrantWhyte's algorithm [18, 19, 20] was tested for comparison of the efficiency of tested methods $\mathcal{F}_3$(or $\mathcal{F}_2$). Fig. 8 shows that both methods were able to build-up the map of unknown environment without any problems.



*Correct shape of the tested environment is plotted as well. $\mathcal{F}_3$ and $\mathcal{F}_2$ strategies provide almost identical results – identical map. Differences are unimportant.

Fig. 8. Environment A – SLAM.

Every method has its own specific characteristics. Leonard-DurrantWhyte's algorithm provides slightly turned map of about 3°. The correct shape of the tested environments was bild by hands for comparison purposes only. In several places there are minor but visible inaccuracies in comparison to the map buit-up by $\mathcal{F}_3$ strategy. In both cases the final line map of the built environment can be used for localization process.

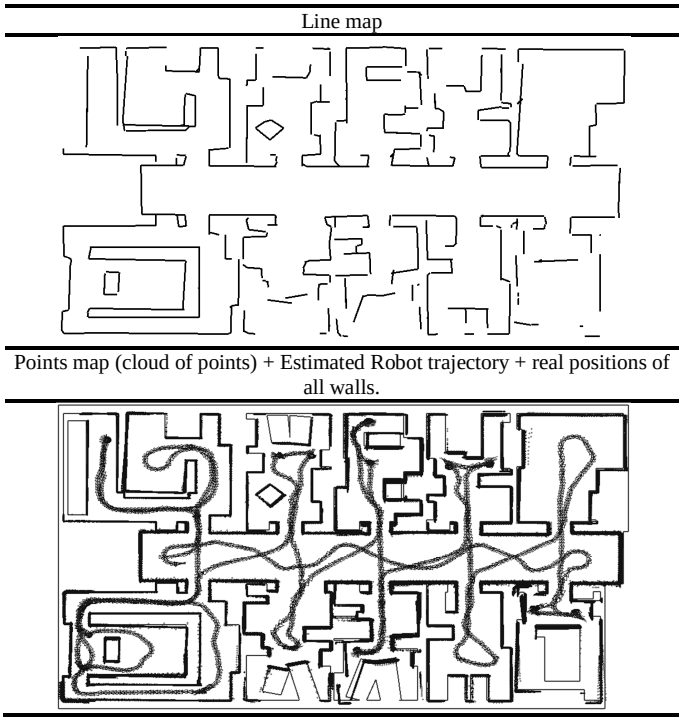


Fig. 9. Environment B - Computer Science Department, University of Bonn

The second experiment was conducted in a large office environment. The trajectory was obtained by computer simulation from the geometrical map of Computer Science Department, University of Bonn. The sensorial data were burdened by additive noise (Gaussian noise, mean zero) with bandwidth 6cm ($\pm$3cm) – it matches the new type of 2DLS Sick LMS-200. Because dimensions of the environment were set to approx. 30x20 meters only, beams were not trimmed by L_{max} function. All beams reflect the walls at any time. It is a big advantage. By this step 2DLS provides more useful information. Fig. 9 shows the result of the experiment.

The robot was able to build the map of environment without any problems. Several places are not well mapped (charted) because of small inaccuracies at heading estimation. Mostly it is small structured space. Thanks to existence of a long corridor in map B, $\mathcal{F}_3$ strategy was only used. Thanks to EA method use, the robot has to keep a minimal distance approx. 50cm from all obstacles – searching area may not overcome the walls of the environment. Sometimes this distance was slightly exceeded. Large and well structured environment is a more suitable area for gSLAM algorithm, especially if there is one central point and all offices are attainable from this. Election of suitable trajectory has a big influence too. The presented gSLAM algorithm only uses one data source – sensorial data from 2DLS. Behavior of such a method is a little different from behavior of a classic probabilistic method. When the robot is moving in a long corridor, the trajectory similar to ship cruising is the best choice. Turning should be made along large circular trajectory, if it is possible. On the other side, gSLAM shows that sensorial data as the only data source can be practically usable for such purposes. Both strategies $\mathcal{F}_3$ and $\mathcal{F}_2$ provide identical results. No malfunction was observed at testing time. Differential evolution provides a stable and very powerful tool.

V. Conclusion

The simultaneous localization and the mapping algorithm were developed and presented in this paper. The core of the presented algorithm is based on geometric primitives and evolutionary computations. Such approach provides an efficient and stable tool. The differential evolution forms a substantial part of this project. Based on practical experiments DE was elected as one of the most suitable algorithms for map building purposes especially on zero or lower noise levels. Results presented in here were obtained from two experiments—in a small indoor office environment and a cluster of small offices and provide us with a wider view on possibilities of the evolutionary robotics and map building process in general.

The proposed algorithm and basic methodology were tested in different types of environments with stable results. Navigation algorithms enabling both, global or local pose estimation and map building (SLAM process) still belong to highly interesting areas of mobile robotics. Constantly increasing computer power provides immense possibilities to create more complicated and more sophisticated algorithms for regular available computers. Thanks to the possibilities of joining the groups of different strategies, great results can be reached regarding to the type of working conditions. The presented map building method only uses one data source and thanks to the natural addition of additive errors at the pose

estimation process, the size of mapped areas will be always limited to some extent.

ACKNOWLEDGMENT

This research was supported by Tanja Agency, to which we would like to express our cordial thanks.

REFERENCES

[1] P.J. Besl, and H.D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239-256, 1992.

[2] J. Brest, S. Greiner, B. Boškovič, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[3] W. Burgard, D. Fox, D. Hennig, and T. Schmidt, "Estimating the absolute position of a mobile robot using position probability grids," *in Proc. of the National Conference on Artificial Intelligence*, pp. 896-901, 1996.

[4] A. Censi, "An ICP variant using a point-to-line metric," *IEEE International Conference on Robotics and Automation ICRA 2008*, pp. 19-25, 2008.

[5] I.J. Cox, "Blanche – An experiment in Guidance and Navigation of an Autonomous Robot Vehicle," *IEEE Trans. on Robotics and Automation*, vol. 7, no. 2, pp. 193-204, 1991.

[6] J.L. Crowley, "World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging," *International Conference on Robotics and Automation*, pp. 674-680, 1989.

[7] J.L. Crowley, "Dynamic World Modeling for an Intelligent Mobile Robot Using a Rotating Ultra-Sonic Ranging Device," tech. report CMU-RI-TR-84-27, Robotics Institute, Carnegie Mellon University, 1984.

[8] J.L. Crowley, "Navigation for an intelligent mobile robot," *IEEE Journal of robotics and automation*, vol. 1, no. 1, 1985.

[9] F. Dellaert, D. Fox, W. Burgard, S. Thrun, "MonteCarlo localization for mobile robots," *Journal of Artificial Intelligence*, vol. 128, no. 1-2, pp. 99-141, 1999.

[10] R.O. Duda, P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley-Interscience, 1976.

[11] M. Ebner, "Evolving environment model for robot localization," *Euro GP 1999*, Ebenhard-Karls-Universitat Tubingen, Germany, Springer Verlag, pp. 184-192, 1999

[12] A.E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transaction on evolutionary computation*, vol. 3, no. 2, pp. 124–141, 1999.

[13] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, pp. 391–427, 1999.

[14] R. Gamperle, S.D. Muller, and P. Koumoutsakos, "A parameter study for differential evolution", *WSEAS Int. Conference on advances in intelligent systems*, pp. 293–298, 2002.

[15] Q. Ke, and T. Kanade, "Robust L1 Norm Factorization in the Presence of Outliers and Missing Data by Alternative Convex Programming," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, 2005.

[16] N.M. Kwok, D.K. Liu, and G. Dissanayake, "Evolutionary computing based mobile robot localization," *Engineering Applications of Artificial Intelligence*, vol. 19, pp. 857–868, 2006.

[17] J.C. Latombe, and A. Lazanas, "Landmark-Based Robot Navigation," *Algoritmica*, vol. 13, no. 5, pp. 472-501, 1997.

[18] J.J. Leonard, and H.F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," *Conference IROS-91*, Osaka, Japan, pp. 1442-1447, 1991.

[19] J.J. Leonard, and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Trans. on Robotics and Automation*, vol. 7, no. 3, pp. 376-382, 1991.

[20] J.J. Leonard, I.J. Cox, and H.F. Durrant-Whyte, "Dynamic map building for an autonomous mobile robot," *Int. journal on Robotics Research*, vol. 11, no./4, pp. 286-298, 1992.

[21] F. Lu, and E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," *Journal of Intelligent Robotics Systems*, vol. 18, no. 3, pp. 249–275, 1997.

[22] E.M. Montes, and A.G.P. Ortiz, "Self-adaptive and Deterministic Parameter Control in Differential Evolution for Constrained Optimization," *IEEE Congress on Evolutionary Computation*, pp. 1375 - 1382, 2009.

[23] H.P. Moravec, and A. Elfes, "High resolution maps from wide angle sonar," *in Proc. IEEE Int. Conf. Robotics and Automation*, pp. 116–121, 1985.

[24] M.R. Mohammadi, and S.S. Ghidary, "Integrated PSO and line based representation approach for SLAM," *in Proceedings of the 2011 ACM Symposium on Applied Computing*, pp.1382-1388, 2011.

[25] L. Moreno, S. Garrido, D. Blanco, and M.L. Muñoz, "Differential evolution solution to the SLAM problem," *Journal of Robotics and Autonomous Systems*, vol. 57, no. 4, 2009.

[26] L. Moreno, J.M. Armingol, S. Garrido, A. Escalera, and M.A. Salichs, "A Genetic Algorithm for Mobile Robot Localization Using Ultrasonic Sensors," *Journal of Intelligent and Robotic Systems*, vol. 34, no. 2, 2002.

[27] T. Pavlidis, and S.L. Horowitz, "Segmentation of Plane Curves," *IEEE Trans. on Computers*, vol. 23, no. 8, pp. 860–870, 1974.

[28] S.T. Pfister, S.I. Roumeliotis, and J.W. Burdick, "Weighted Line Fitting Algorithms for Mobile Robot Map Building and Efficient Data Representation," *ICRA 2003*, pp. 14-19, 2003.

[29] S.T. Pfister, "Weighted line fitting and merging," Tech. Rep., California Institute of Technology (2002), Available: http://robotics.caltech.edu/~sam/TechReports/LineFit /linefit.pdf

[30] A.K. Qin, and P.N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," *The 2005 IEEE Congress on In Evolutionary Computation*, vol. 2, pp. 1785-1791, 2005.

[31] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut, and R Husson, "An Optimized Segmentation Method for a 2D Laser-Scanner Applied to Mobile Robot Navigation," in *Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications*, 1997.

[32] M. Schmidt, "Least Squares Optimization with L1-Norm Regularization," CS542B Project Report, 2005.

[33] P. Skrzypcynsky, "Simultaneous Localization and mapping: a feature based probabilistic approach," *Int. Journal of Applied Mathemahics and Computer Science*, vol. 19, pp. 575-588, 2009.

[34] P. Skrzypczyński, "Building Geometrical map of environment using IR range finder data," *Intelligent Autonomous Systems*, U.Rembold et. al. IOS Press, 1995.

[35] R. Storn, "On the Usage of Differential Evolution for Function Optimization," *NAFIPS'96*, pp. 519–523, 1996.

[36] R. Storn, and K. Price, "Differential Evolution – a Simple and Efficient Heuristic for Global Optimization," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[37] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Journal of Soft Computing*, vol. 10, pp. 673–686, 2006.

[38] C.K. Wong, J. Schmidt, and W.K. Yeap, "Using a Mobile Robot for Cognitive Mapping," *International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007.

[39] W.K. Yeap, "Towards a computational theory of cognitive maps," *Artificial Intelligence*, vol. 34, pp. 297-360, 1988.

[40] R. Kummerle, G. Grisetti, H. Strasdat, K.Konolige, and W. Burgard, „g2o: A General Framework for Graph Optimization,", in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3607-3613, 2011.

[41] N. Metropolis, and S. Ulam, "The Monte Carlo method," *Journal of the American Statistical Aassociation*, vol. 44, 335-341, 1949.

[42] D. Whitley, S. Rana, and R.B. Heckendorn, „The Island Model Genetic Algorithm: On Separability, Population Size and Convergence," *Journal of Computing and Information Technology*, vol. 7, pp. 33-47, 1998.

[43] R. Tanese, „Distributed Genetic Algorithms," *Proc. of the Third International Conference on Genetic Algorithms*, MorganKaufmann, J.D.Schaffer editor, pp. 434-439, 1989.

[44] M. Gorges-Schleuter, "Explicit parallelism of Genetic Algorithms through Population Structures," *Paraller Problem Solving from nature*, Springer Verlag, H.P. Schwefel and Reinhard Manner, editors, pp. 150-159, 1991.

[45] M. Begum, G.K.I. Mann, and R.G. Gosine, "An Evolutionary Algorithm for Simultaneous Localization and Mapping of Mobile Robots," *Proc. of The IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pp. 4066-4071, 2006.

[46] M. Begum, G.K.I. Mann, and R.G. Gosine, „A Fuzzy-Evolutionary Algorithm for Simultaneous Localization and Mapping of Mobile Robots," *Proc. of IEEE Congress on Evolutionary Computation*, pp. 1975-1982, 2006.

[47] M. Begum, G.K.I. Mann, and R.G. Gosine, „Intergrated fuzzy logic and genetic algorithmic approach for simultaneous localization and mapping of mobile robots," *Journal of Applied Soft Computing*, vol. 8, no. 1, January, 2008.

[48] D.E. Goldberg, "Simple genetic algorithms and the minimal deceptive problem," *Genetic Algorithms and Simulated Annealing, London*, Pitman, pp. 74-88, 1987.

[49] D.E. Goldberg, "*Genetic algorithms in search, optimization, and machine learning*," Addison-Wesley New York, 1989.

[50] K. Price, D. Corne, M. Dorigo, and F. Glover, "An Introduction to Differential Evolution," Eds. London, McGraw-Hill, pp. 79–108, 1989.

[51] K. Price, and R. Storn, "Minimizing the Real Functions of the ICEC'96 contest by Differential Evolution," *IEEE Int. Conference on Evolutionary Computation (ICEC'96)*, pp. 842–844, 1996.

[52] M. Dorigo, "Optimization, Learning and Natural Algorithms", Ph.D. dissertation, Politecnico di Milano, Italy, 1992.

[53] J. Kennedy, and R.C. Eberhart, "Particle swarm optimization," *Proc. of the 1995 IEEE International Conference on Neural Networks*, 1995.

[54] J. Moravec, "Cascaded Evolutionary Estimator for Robot Localization," *International Journal of Applied Evolutionary Computation (IJAEC)*, vol. 3, no. 3, pp. 33-61, 2012.

[55] J. Moravec, "Continuous Robot Localization in Known Environment Using Genetic Algorithms," *The 10th IEEE Int. Conference on Fuzzy Systems FUZZ IEEE 2001*, Melbourne, Australia, p.6, 2001