International Journal of Robotics and Automation (IJRA)
Vol. 1, No. 1, March 2012, pp. 49~63
ISSN: 2089-4872

Reduced Search Space Algorithm for Simultaneous Localization and Mapping in Mobile Robots

H. Omranpour *, S. Shiry *
* Departement of Computer Engineering, *Amirkabir University of Technology*, Tehran, Iran.

Article Info

Article history:

Received Feb 13, 2012
Revised Mar 14, 2012
Accepted Mar 23, 2012

Keyword:

Evolutionary algorithm
Fitness function
Pareto
Particle swarm optimization
Search space
SLAM

ABSTRACT

In this paper, we propose a new algorithm for simultaneous localization and mapping in mobile robots which uses evolutionary algorithm and particle swarm optimization. The proposed method is based on both local and global heuristic search methods. In each step of robot movements, the local search is applied in the small search space of odometry errors to improve the map accuracy. A global search method is applied for loop closing. The proposed algorithm detects loops and closes them, detects and solves correspondence and avoids local extremums. With a proper representation of problem parameters in chromosome, the dimensionality of search space is reduced. The proposed algorithm utilizes occupancy grid and does not require land marks which are not available in most natural environments. A new fitness function is proposed that is computationally efficient and eliminates the need for complex statistical calculations as used in current approaches. Results of experiments on real datasets exhibit the superior performance of the proposed method compared to the current methods.

Corresponding Author:

S. Shiry
Departement of Computer Engineering, *Amirkabir University of Technology*, Tehran, Iran.
Email: Shiry@aut.ac.ir

1. INTRODUCTION

Rapid progresses in autonomous mobile robots enable them to autonomously explore unknown environments of other planets or ocean floors. Because there is no prior knowledge about these environments available to the robots, they must be able to build the environment map online and to localize themselves on the map concurrently.

Map building is to acquire a model of the surrounding of the robot and localization is to identify the location and state of the robot in the obtained model. Maps are used for robot localization and navigation. To build a map, robot must be equipped with sensors like camera, sonar, laser, infrared, radar, touch sensor, compass and Global Positioning System. Because most of these sensors have limited range and measurement error, the robot must explore the environment to build a complete map. In 80s and early 90s, map building methods were classified into topologic and metric categories. An example of metric methods is occupancy grid which represents the map with a network of empty or occupied cells. Topologic methods display the environment with the use of a list of special locations which are connected together with a set of edges. These edges contain information about how the robot navigates among different places [1].

In order to navigate autonomously and intelligently, a mobile robot must have the environment map and its location on it. In recent years, simultaneous localization and mapping (SLAM) methods are developed to provide a solution for the need of concurrent localization of robot and other important objects in the environment [2].

In SLAM, mobile robot receives information about the environment from its sensors, processes them, builds a correct map and localizes itself to autonomously explore the environment [3].

Several methods have been proposed to solve this problem which will be discussed in following section. In this paper, we propose a new intelligent method for simultaneous localization and mapping which minimizes error in map and path. Because of their desired features in this problem, we have used evolutionary algorithm and particle swarm optimization.

In section 2, we explain simultaneous localization and mapping and review current solutions to this problem. In section 3 we describe the proposed algorithm and compare its features to other methods. Section 4 is dedicated to exhibit the results of experiments. In section 5 we discuss the ideas in the proposed method and section 6 is the concluding remarks.

2. SIMULTANEOUS LOCALIZATION AND MAPPING

The problems of localization and mapping for robot depend on each other; it means that if the robot location is known then map building would be easy. Also if the map is known, there are many efficient algorithms to determine the robot location. But solving both problems simultaneously is very difficult [1].

A solution to SLAM must deal with the following problems [1,4,5]:

Sensor uncertainty: One of the most important problems in SLAM is measurement error of robot odometry which results in robot localization uncertainty. In modeling problems, if different measurement errors are statistically independent, it would be easy to resolve them. However in map building for robot, these errors are statistically dependent. This is because control error accumulates over time which leads in more error in interpretation of data for subsequent measurements. Fig.1 illustrates this problem. Resolving these errors is important to build a correct map.

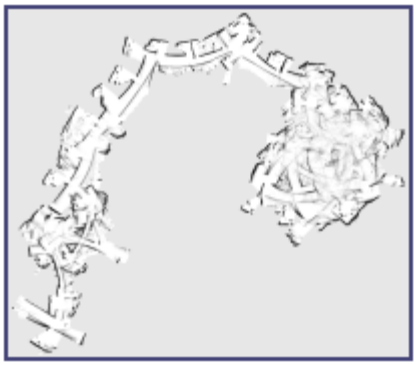

Fig.1 A map generated using laser and odometry with measurement errors [27]

Fig.2 The actual map in Fig.1 [27]

Correspondence problem: Another main problem in map generation is the correspondence problem. Detecting correspondence means to recognize that different sensory measurements belong to a unique physical object.

Loop closing: A solution to this problem aims to detect and close loops in map with the use of correspondence detection. When the robot reaches the end of a loop, it must locate itself in the map it has built so far. This is difficult because the accumulated measurement errors might be enormous. Another difficulty arises because the number of hypothetical maps and robot locations grow exponentially with time. Fig.2 represents the map in Fig.1 with the aforementioned problems solved.

Time complexity: computational efficiency is an important issue in the design of algorithms for autonomous mobile robots because they have limited computational resources. An algorithm proposed for SLAM must guarantee real-time performance on these limited set of resources.

Memory complexity: memory usage is too an important problem in algorithms developed for mobile robots. A memory consuming process can prevent other processes from execution and therefore suspend robot activities.

2.1 Related Works

Several methods have been proposed to solve the SLAM problem. Extended Kalman Filter (EKF), Fast SLAM and Distributed Particle (DP) SLAM are the most popular ones. EKF eliminated the linear motion model requirement in Kalman Filter (KF), and produces accurate results. However it fails in environments where error model is not Gaussian and it is computationally inefficient [6-10]. Fast SLAM combines the particle filter [11-13] and KF methods to improve the efficiency of KF method. Each particle stores the robot trajectory and map to increase the speed of algorithm. However, storing map in each particle requires a large amount of memory. Rao–Blackwellized particle filter method [14,15] proposed a mechanism to share the maps between particles to compensate memory requirement of Fast SLAM, but requires predetermined landmarks in the environment.

DP-Slam uses a complex data structure to represent multiple particles in a single map. In this method a occupancy map is used which cells are trees that represent map parts of different particles. This data structure reduces the memory requirement but put a burden on speed of algorithm due to data manipulation complexity. This algorithm does not require landmarks and its correspondence detection is not accurate [16,17].

In [18], particle filter is combined with Genetic algorithms to alleviate the problem of local minima. This method uses image processing for correspondence detection which acts very well, however this puts a burden on processing time of the algorithm. Begum et.al. [5] used a set of fuzzy rules to detect errors in sensory measurements. It is based on Genetic algorithms and uses a fitness function with a complicated formula for matching. The method used the Island Genetic algorithms (IGA) to increase speed and variety. The IGA is fast if the number of processors is equal to the number of islands. However, an autonomous mobile robot usually has a single processor. In this case, IGA performance is like simple GA and therefore speed decreases. Moreover, the memory requirement for IGA is more than simple GA.

We reduced the search space to increase the speed of finding optimal solution in the evolutionary algorithm. This reduction also lowers the memory requirements of the proposed algorithm.

3. PROPOSED ALGORITHM

The problem of SLAM can be defined as an optimization in the space of all possible maps and robot trajectories [19]. However the search space of maps and paths are extremely large because they are high dimensional data structures. Search in this large space for optimum solutions is very time consuming [5, 19]. In order to decrease the number of dimensions in search space, we quantized the error measurements into $L<<M$ partitions where M is the total number of sensory measurements along the robot trajectory. Optimization is performed on each robot step and on the set of all partitions which both have much smaller number of dimensions. Optimization on each robot step increases the convergence speed of the optimization on all partitions.

The algorithm we propose here is based on two search methods (Fig.3). One is a local search algorithm that uses particle swarm optimization. The goal of local search is to extend and improve the map and robot path efficiently and in real-time, using current map and sensory information. Global search algorithm is used to solve correspondence and loop closing problems. The algorithm detects correspondence or loop and then performs a global search.

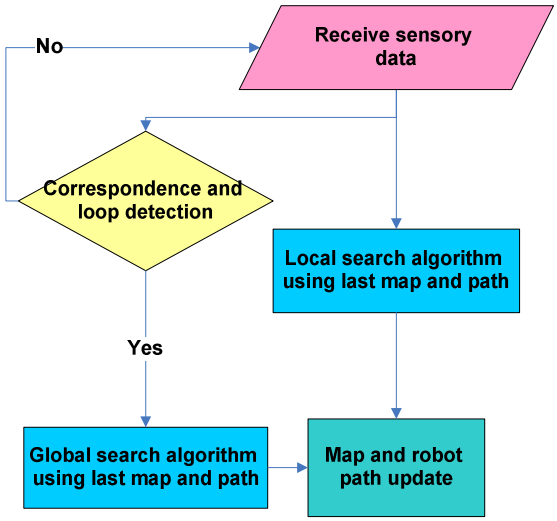

Fig.3 Flow chart of the proposed algorithm

The evaluation functions we used here are based on combination of localization and mapping information. Map representation model is important because it influences the time and memory complexity of algorithm. We used occupancy grid to represent the map because of its simplicity in structure and use, moreover it is fast in processing homogeneous information. Another reason to use occupancy grid is that our proposed algorithm does not require landmarks.

3.1 Local search algorithm

Local optimization is performed to increase the accuracy of odometry measurements in each robot step. Locally accurate maps and robot poses are then grouped together by the global search algorithm to find the globally accurate map and path with high speed. In order to perform optimization efficiently and with low memory requirements, and also to keep the size of search space small in each step, we used particle swarm optimization (PSO).

PSO is a stochastic optimization technique based on solutions of a population of particles. Inspired by social behaviors of bird flocking and fish schooling, it was first introduced by Kennedy and Eberhart in 1995 [20].

This method is based on a set of particles which move in search space. Each particle consists of a location (p[]) and a velocity (v[]) vector. It also retains the best observed solution of itself (pbest[]) and of its group (gbest[]). Each particle moves with appropriate velocity toward the best found solution of itself and its group to find the optimum solution. PSO can find acceptable answers to difficult nonlinear and discrete optimization problems [21].

In the proposed algorithm, we used each particle to represent accumulated error in robot movement. Δx, Δy and $\Delta\Theta$ represent the amount of error in location (X,Y) and orientation Θ, respectively for the last robot step. As the robot moves, these errors are calculated in each step and added to sensory measurements to localize the robot and generate the map correctly. In this problem, the search space is the cube of [-maxΔx, maxΔx], [-maxΔy, maxΔy] and [-max$\Delta\Theta$, max$\Delta\Theta$] in (x, y, Θ) space. This is a small search space because measurement error in each step lies in a tight range (Fig.4). Therefore we can find answer quickly by applying the PSO Algorithm. First we distribute the particles in a small search cube, and then the acceptable solution is found as the particles move in this space.

In order to calculate the evaluation function in real-time, we used Eq. 1.

$$evaluation_1(t) = -\sum_{i=1}^{n_t}\sum_{j=1}^{m_t}\begin{cases}1 & \text{if}(map_{[part(t)]}(i,j) > 0)\\ 0 & \text{if}(map_{[part(t)]}(i,j) = 0)\end{cases} \quad (1)$$

'map[part(t)]' is part of the map that is being updated currently and mt and nt represent its dimensions. Every time an obstacle is detected in the location (i,j) of the map, the corresponding cell value is increased by 1. The evaluation function of (Eq.1) calculates the number of observed obstacles in the map.

Whatever these obstacles overlap and decrease in number, the obtained map will be more accurate. We seek to find the maximum of the evaluation function; therefore a negative sign is applied in (Eq.1).

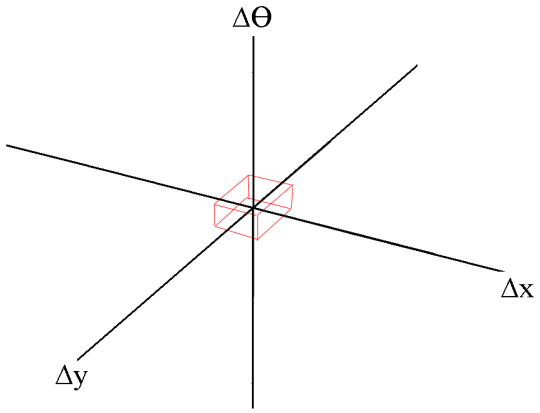

Fig.4 Search space for local search algorithm is confined to the cube. This is a small search space because measurement errors in each step are small

We have tried a second evaluation function which calculates the mean of the number of observed obstacle cells (Eq. 2).

$$evaluation_{\ 2}(t) = \frac{\sum_{i=1}^{n_t}\sum_{j=1}^{m_t} map_{[\,part\,(t)]}(i,j)}{\sum_{i=1}^{n_t}\sum_{j=1}^{m_t}\begin{cases}1 & \text{if}(map_{[\,part\,(t)]}(i,j)>0)\\ 0 & \text{if}(map_{[\,part\,(t)]}(i,j)=0)\end{cases}} \quad (2)$$

Initially a population of random particles is generated. Moving the particles of population, we seek to find the optimum solution.

In each iteration, pbest and gbest are determined. Then we use Eq. 3 to calculate the velocity and Eq. 4 to update the position of each particle.

$$v_{i,t} = w.v_{i,t-1} + c_1.r_1.(pbest_{\ i} - p_{i,t}) + c_2.r_2.(gbest_{\ i} - p_{i,t}) \quad (3)$$

$$p_{i,t+1} = p_{i,t} + v_{i,t} \quad (4)$$

Here, r_1 and r_2 are random numbers in the interval [0,1], c_1 and c_2 are learning rates usually set to 2.

Because time order of PSO algorithm is constant and it does not require evaluation of complicated mathematical expressions [22], this algorithm can seek local search space and find the optimal solution in real-time with few particles and movements.

3.2 Global Search Algorithm

The goal of the global search is to detect correspondences and close loops. If the robot encounters a place for the second time but the laser sensor can not detect obstacles in their right location then loop or correspondence is detected and the global search algorithm is called.

The global search is an evolutionary algorithm which aims to find the minimum error of poses to build a consistent map. Contrary to the local search, the global search algorithm uses a sequence of robot poses and laser sensor measurements to update a large portion of the map. A problem that arises here is that the dimensionality of search space is proportional to the number of robot steps along the path. In order to reduce the dimensionality, we divided these steps into a small number of groups. Error in poses and map of

each group is represented with a vector in chromosome. These errors are multiplied to a Gaussian function and are distributed over group steps to avoid loosing precision.

Evolutionary algorithms are a class of search algorithms inspired by evolution of animates. The most important advantage of evolutionary algorithms is that they are applicable to a wide range of problems with large space of possible solutions. With proper use of evolutionary operators, they can pass local extremums and find the global extremum [23].

Each evolutionary algorithm is composed of eight parts which must be designed carefully to meet the specific problem requirements. We will consider each part for the proposed algorithm.

3.2.1 Representation

The first step in using an evolutionary algorithm is to appropriately code the parameters of problem in a format suitable for the algorithm. In this section, we code the important parameters of the problem in persons of population which are called chromosome.

We assume that the data of M successive time steps are available to this algorithm. As measurement error is a continuous function, we divide this data into L parts. Therefore each chromosome becomes a 3*L*2 matrix. We chose L<<M to speed up the algorithm and reduce the memory requirements. Each column in the chromosome represents an error part. Errors of location in (x,y) coordinate and direction in the i-th part are represented by Δxi, Δyi and ΔΘi respectively. These errors are weighted with a Gaussian function and added to the i-th part of the robot locations and therefore to the obstacles locations. Error of i-th part for x, y and Θ axis are multiplies to a Gaussian functions with means μxi, μyi and μΘi. The chromosome with L parts is represented in Fig.5.

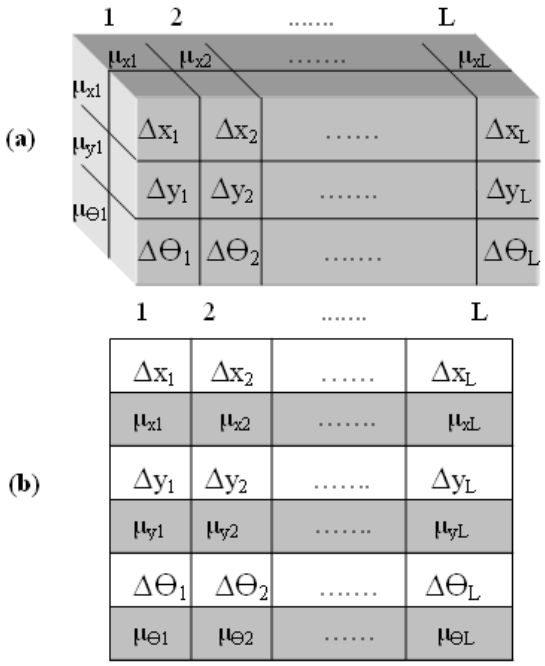

Fig.5 (a) 3D representation of chromosome, (b) 2D representation of chromosome

Each obtained column of error is added to the corresponding part using a Gaussian weighting function. This is shown for x coordinate and i-th part in Eq. 5.

$$error_{i,j}^{x} = \Delta x_i \times \frac{1}{\delta_x \sqrt{2\pi}} \exp\left(-\frac{(path_{i,j}^{x} - \mu_{xi})^2}{2\delta_x^{\,2}}\right) \quad (5)$$

Where $error_{i,j}^{x}$ represents the obtained error and $path_{i,j}^{x}$ is the index in the range $\left[1, \frac{M}{L}\right]$ of j-th measurement in the i-th part for x axis. Standard deviations for x, y and Θ depend on the model of environment and robot sensors. We have considered them to be constant for this experiment. To avoid loosing precision because of dividing measurements to L parts, we used a similar Gaussian weighting mechanism with an appropriately selected mean to increase the speed and accuracy of the algorithm.

3.2.2 Initialization

There is no information available about the pattern of solution to this problem. Therefore random numbers of uniform distribution in a range between minimum and maximum values of the measurement errors and mean are used to initialize the population.

3.2.3. Fitness Function

In order to select parents and survivors, we must evaluate chromosomes in the population. The goal of the evolutionary algorithm is to find the maximum (minimum) of the fitness function. We combined two different fitness functions in Eq. 6 and Eq. 7 to evaluate the population.

$$fitness_1 = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} map(i,j)}{\sum_{i=1}^{n}\sum_{j=1}^{m}\begin{cases}1 & \text{if}(map(i,j)>0)\\ 0 & \text{if}(map(i,j)=0)\end{cases}} \tag{6}$$

$$fitness_2 = -W_1.\sum_{i=1}^{n}\sum_{j=1}^{m}\begin{cases}1 & \text{if } ((map(i,j)>0)\,\&\,(\text{traj(i, j)} == 1))\\ 0 & else\end{cases} \tag{7}$$
$$-W_2.Dis\tan ce(obj(t_1), obj(t_2))$$

In Eq. 6, sum of observations of obstacles in each map cell is divided to the number of occupied cells to calculate the mean of the times an obstacle is observed in a cell. A weighted sum of two terms is calculated in Eq. 7. The first term with weight –W1, counts the number of points in the robot path which have conflict with the observed obstacles in the map. The second term, measures the distance between two objects in the map where a correspondence has been detected among them. The W1 and W2 are used to adjust the effect of each term. Their values depend on the specific problem in hand. In order to avoid the undesirable effect of relative magnitude of each fitness function on the other, we used Pareto's method to maximize both concurrently [24].

3.2.4 Parent selection

Considering chromosome representation, we preferred to select parents in random, because it is possible to combine a good and a poor solution and generate a better one. Even combining two poor solutions might result in an offspring with high fitness value. This increases the population variety and extends search space.

3.2.5 Crossover

To improve the population variety, a uniform crossover is used. Two parents are selected in random and combined with probability Pc. The odd and even columns are selected from each parent and interleaved to generate two children. This operation is repeated to produce a predetermined number of offspring. Each column of the chromosome in Fig.5 contains 3 pairs of numbers. Each pair represents error and mean for x, y and Θ dimensions. The proposed crossover operator is designed to ensure the variety and convergence speed of the global search algorithm. In order to achieve variety, in the first half of the generations each number is selected with the probability 0.5 from a parent. To increase the convergence speed, in the second half of the generations, each pair of numbers comes from a parent with probability 0.5 (Fig.6). Error and mean in each of x, y and Θ dimensions are selected together to prevent changes in error effect on measurement parts. This inhibits generating incompetent offspring and speeds convergence to optimum solution.

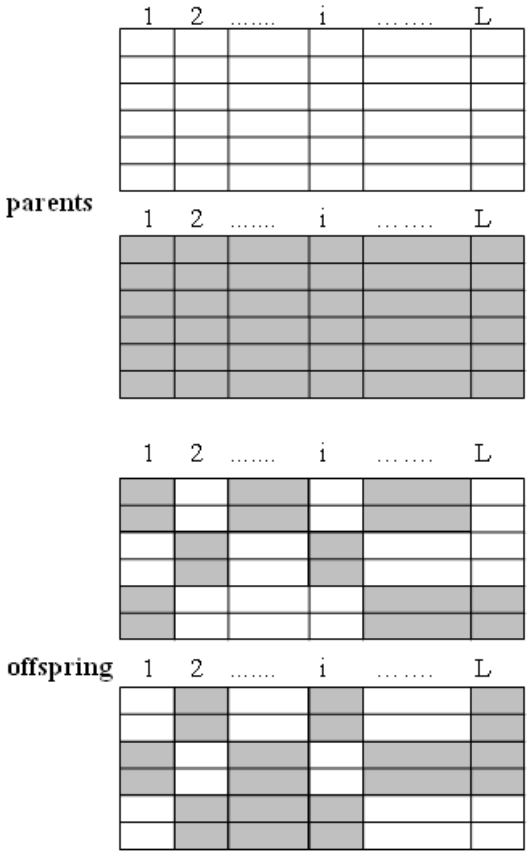

Fig.6 Uniform crossover used in the second half of generations in global search algorithm

3.6.2 Mutation

This is an important operator for generating new populations in intact portions of the search space. It also can help search algorithm to pass over local extremums [25].

Mutation takes place for each offspring column with probability Pm (Fig.7). Each one of the three pairs in chromosome column is selected with probability 1/3. A random number in the range [-c.maxΔx, c.maxΔx] or [-c.M/L,c.M/L] is added to one of the error values or μ correspondingly, if it does not violate the valid range. c is a small constant e.g. 0.1.

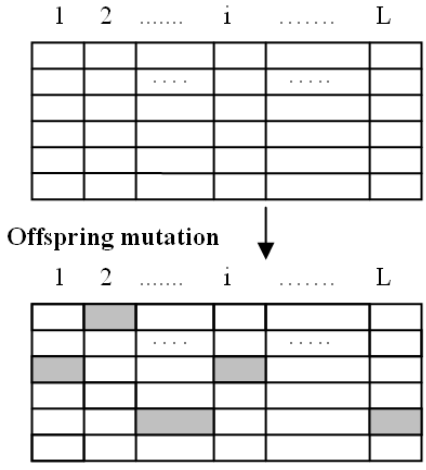

Fig.7 Mutation operator in global search algorithm

For mutation, it is probable for all values in the chromosome to change. All values of error in the valid range are explored by changing Δx, Δy and $\Delta\Theta$. Changes in the corresponding μ are admitted to reach a better error distribution.

Mutation is an important operator. A search algorithm using only crossover cannot find the optimum solution but with mutation alone it is possible. The reason is that crossover searches a space confined to current parents but mutation is able to generate populations in all points of search space.

3.2.7 Survivors selection

This method is based on tournament and $\mu+\lambda$ where μ is the population size and λ is the offspring number. Advantage of this selection is its high speed and guaranteed convergence to global or local optimum. Pareto's method is used in survivors selection. Eighty percent of survivor population is selected from offspring and the remaining 20% comes from parents. In tournament, q chromosomes are selected randomly and the best of them is chosen to remain in the next generation. To select μ survivors, tournament is repeated for 80% of population size (μ) in the offspring population and for 20% of μ in the parent population. Another benefit of this method is to select proper persons from previous population that results in diversity and convergence ability to reach optimum solution [26].

3.2.8 Termination condition

We used disjunctive combination of two conditions to reach the solution quickly and with high precision, until the number of generations reaches a maximum value or variance of the fitness reaches a lower threshold.

If correspondence or loop is not detected after k time steps (Fig.3), the global search algorithm is called because detection algorithm might work improperly. In this case, W2 in Eq. 7 is set to 0.

3.3 Comparison with other methods

In this section we compare the properties of the proposed method with current methods (table.1).

Table.1 Comparing proposed method with other methods

EKF	Fast slam	DP-slam	Proposed Method	
Yes	Yes	No	No	Landmark
High	High	Med	High	Loop closing and correspondence
High	Low	High	Med	Time complexity
No	No	Yes	No	Local minima
High	High	Med	Low	Memory complexity
Yes	Yes	Yes	Yes	Convergence
Gaussian	Any	Any	Any	Error model

Each row compares one feature of all methods. The last row indicates that error model which entirely misleads the robot path generation is arbitrary in the proposed method. It is clear from Table. 1 that the proposed method is superior in features to other mentioned methods. For example the time required to process map in Fig. 8.b for the proposed algorithm is ½ of the time required for DP-SLAM method.

4. EXPERIMENTAL RESULTS

We used two datasets to demonstrate the results of proposed method [27]. Primary map and path trajectory with odometry error is shown in Fig. 8.

By selecting 10 particles and applying the local search algorithm for the first loop of Fig.8 maps and paths shown in Fig.9 are generated after 50 movements.

Global search algorithm is called after a correspondence or loop has been detected. Results of this algorithm are shown in Fig.10. In global search algorithm the population size is 10, number of generations is 100, size of children is $7*\mu$ and q in tournament selection is set to 5. Pm and Pc are set to 0.05 and 0.8 correspondingly. These values are selected to maximize the convergence of experiments to optimal solution. It is obvious in Fig.10 that the global search algorithm has correctly closed the loops.

(a)

(b)

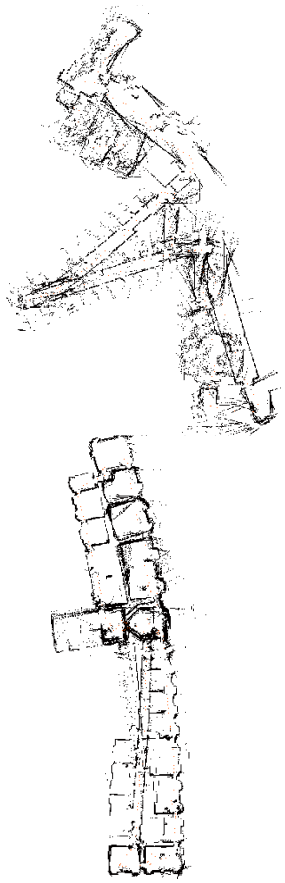

Fig.8 Maps and trajectory used in experiments

(a)

(b)

Fig.9 Results of the local search algorithm for map in Fig.8.a (a) using Eq.1 and (b) using Eq. 2 as evaluation function

Convergence of global search algorithm to optimal solution is show in Fig.11 for two fitness functions in Eq. 6 and Eq. 7 from Fig 8.b.

(a)

(b)

Fig.10 Results of the global search algorithm (a) for first loop of Fig 8.a and (b) for Fig 8.b

(a)

(b)

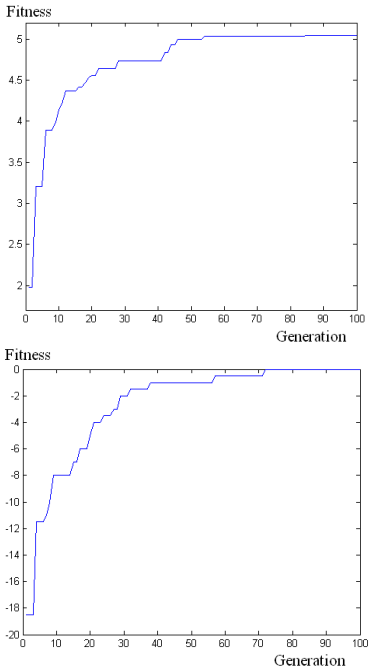

Fig.11 Best fitness function for 100 generations for map in Fig 8.b (a) using Eq. 6 (b) using Eq. 7. W1 and W2 are set to 0.5

It can be seen in Fig.11 that the global search algorithm approaches the optimal solution in an acceptable generation numbers.

5. DISCUSSION

In this section we discuss the novel ideas proposed in this paper. The goal of evaluation function of Eq.1 is to increase overlap in the points of obstacles in the map for accurate map generation. In Fig.12, black lines display the observed obstacles by robot using laser sensor and the red arrow is the state of the robot in time step i. Because of odometry error, robot observes the obstacles in step i+1 like the blue lines in Fig12.b. The goal is to correct the robot state which leads in obstacle map shown in blue lines of Fig12.c.

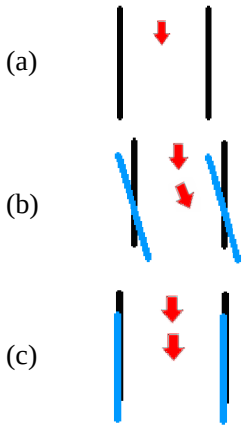

Fig.12 (a) Observed obstacles and state of robot in time step i, (b) Observed obstacles and state of robot with odometry error in time step i (black) and i+1 (blue), (c) Correct state and obstacles in (b)

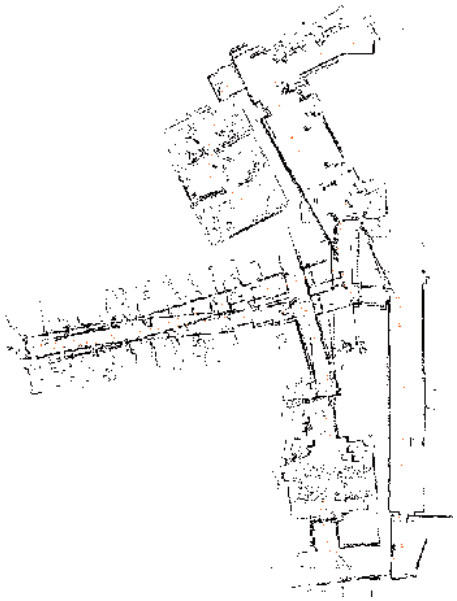

Fig.13 Map and path generated by global search algorithm with Eq. 6

An evaluation function for this goal should give credit to states with overlap in observed obstacles. In Fig.12.b the number of obstacles in occupancy grid is large and they overlap with previous observations in few points, unlike points in Fig.12.c. The local search algorithm with evaluation function of Eq.1 finds the best state (x,y,Θ) for robot in which the number of obstacles in map is smallest and these obstacles highly overlap. The second evaluation function in Eq. 2 calculates the average number of observations for each cell

that is detected as obstacle. The resulting map is more accurate with maximal overlap and higher values of average number of occupation for each cell.

Time complexity of local search algorithm with k scanned obstacles, m particles and n iterations belongs to Θ(kmn). If we use free cells in evaluation functions, with c free cells between the robot and obstacles, the time complexity becomes Θ((c+1)kmn). However evaluation functions are designed to be independent of free cells to increase speed of the evolution and algorithm so that it can be used online.

The fitness function in Eq. 6 is similar to Eq. 2, with the calculations over the entire map. Because the global search algorithm is designed to close loops and correspondences, another fitness function was suggested in Eq. 7. If we use only the fitness function in Eq.6, it may generate maps similar to Fig.13. In order to avoid conflict between path and obstacles, we proposed the fitness function in Eq. 7. This function uses two terms; the number of conflict between path and obstacles and distance between two observations of a single object. The global search algorithm using fitness function of Eq. 7 with only first term generates map like Fig.14. This is because the map is extended to avoid conflicts.

Fig.14 Map and path generated by global search algorithm with first term of Eq. 7. The map is extended to avoid conflicts

If we use the two terms in Eq. 7 summed with the fitness function in Eq. 6 we acquire the map in Fig.15 which is well but not perfect. In order to get the best possible solution, we used Pareto's method instead of simple addition to combine two fitness functions. The result is shown in Fig.16.

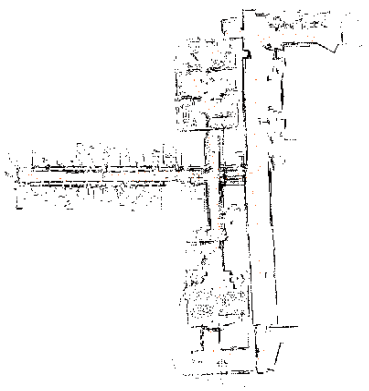

Fig.15 Map and path generated by global search algorithm with fitness using sum of Eq. 6 and Eq.7

In the global search algorithm, if each person requires d units of memory for each of its cells, length of chromosome is l, population size is μ, number of children is λ and memory to store a map is M then memory requirement belongs to Θ(6dl(μ+λ)+M). This is independent of the number of generations. The algorithm was designed to avoid keeping multiple maps and uses one map in memory during its execution. Therefore the algorithm is memory efficient.

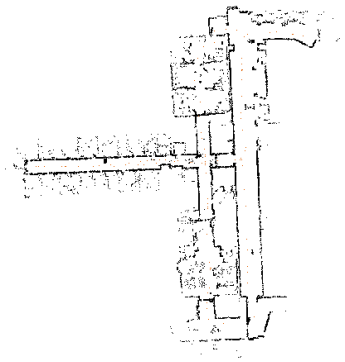

Fig.16 Map and path generated in global search algorithm with Pareto's method

6. CONCLUSION

Simultaneous localization and mapping is an important problem in robotics. In this paper, with an optimization approach, we used evolutionary algorithm and particle swarm optimization to solve the SLAM problem. With suitable representation of problem parameters in particles and a novel evaluation function, the local search algorithm could solve the problem and find the solution with high accuracy and speed. To overcome the loop closing and correspondence problems, we used a memory efficient algorithm with a heuristic fitness for global search in which Pareto's method was used to combine two fitness functions. With a novel representation of parameters in chromosome, we reduced the dimension of search space.

The proposed method has desirable features, like no need for landmarks in the environment. Results of experiments on real datasets revealed that the proposed algorithm can be implemented and used on real robots efficiently. We used only raw laser data which is inefficient for detecting correspondence and loops. Future works should address the problems of loop and correspondence detection and developing fitness functions.

References

[1] Thrun, S.:Robotic mapping: a survey, Technical Report CMU-CS-02-111, CMU, PA, (2002)
[2] Nieto, J., Bailey, T., Nebot, E.: Recursive scan-matching SLAM. Robotics Autonomous Systems, 66(1), 39–49 (2007)
[3] Martinelli, A., Nguyen, V., Tomatis, N., Siegwart, R.: A relative map approach to SLAM based on shift and rotation invariants. Robotics Autonomous Syst. 55(1), 50–61 (2007)
[4] Frese, U.: A discussion of simultaneous localization and mapping. Autonomous Robots, 20, 25–42 (2006)
[5] Begum, M.,. Mann, G., Gosine, R.G.: Integrated fuzzy logic and genetic algorithmic approach for simultaneous localization and mapping of mobile robots. Applied Soft Computing Journal, 8(1), 150-165 (2008)
[6] Smith, R., Self, M.P., Cheeseman, P.: Estimating uncertain spatial relationships in robotics. Auton. Robot Vehicles, 167–193 (1990)
[7] Dissanayake, M.W.M.G., Newman, P., Clark, S., Durrant-Whyte, H.F., Csorba, M.: A solution to the simultaneous localization and map building (SLAM) problem. IEEE Trans. Robot. Automat. 17(3), 229–241 (2001)
[8] Newman, P.: On the structure and solution of the simultaneous localization and map building problem. Ph.D. Thesis, University of Sydney, (2000)
[9] Durrant-Whyte, H.: An autonomous guided vehicle for cargo handling applications. Int. J. Robot. Res, 15(5), 407–440 (1996)
[10] Leonard, J.J., Durrant-Whye, H.F.: Mobile robot localization by tracking geometric beacons. IEEE Trans. Robotics Automation, 7(3), 376-382 (1991)
[11] Montemerlo, M.: Fast SLAM: A factored solution to the simultaneous localization and mapping problem with unknown data association. Ph.D. Thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. (2003)
[12] Roller, D., Montemerlo, M., Thrun, S., Wegbreit, B.: Fastslam 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. Proceedings of the International Joint Conference on Artificial Intelligence. (2003)
[13] Ristic, B., Arulampalam, S., Gordon, N.: Beyond the Kalman Filter: Particle Filters for Tracking Application. Artech House. (2004)
[14] Grisetti, G., Stachniss, C., Burgard, W.: Improving grid-based slam with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, Barcelona, Spain, pp. 2443–2448 (2005)

[15] Grisetti, G., Tipaldi, G.D., Stachniss, C., Burgard, W., Nardi, D.: Fast and accurate SLAM with Rao-Blackwellized particle filters. Robotics Autonomous Syst., 55(1), 30-38 (2007)
[16] Eliazar, A., Parr, R.: DP-SLAM: fast, robust simultaneous localization and mapping without predetermined landmarks. Proceedings of the 8th Int'l Joint Conference on Artificial Intelligence, pp. 1135-1142 (2003)
[17] Eliazar, A., Parr, R.: DP-SLAM 2.0, Proc. IEEE Int. Conf. Robotics Automation. 2, 1314-1320 (2004)
[18] Kwok, N.M., Rad, A.B.: A modified particle filter for simultaneous localization and mapping. J. Intell. Robot Syst., 46(4), 365-382 (2006)
[19] Ducket, T.: A genetic algorithm for simultaneous localization and mapping. Proceedings of the IEEE International Conference on Robotics and Automation, 1, 434–438 (2003)
[20] Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. Proceedings 6th International Symposium Micromachine Human Science, Nagoya, Japan, 39–43 (1995)
[21] Kennedy, J., Eberhart, R., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco, (2001)
[22] Shi, Y., Eberhart, R.: Fuzzy adaptive particle swarm optimization. IEEE Evolutionary Computation, 1, 101-106 (2001)
[23] Koza, J. R.: Genetic Programming: On the Programming of Computers by Means of Nature Selection. MIT Press, Cambridge, MA, (1992)
[24] Agrawal, S., Dashora, Y., Tiwari, M.K., Son, Y.J.: Interactive particle swarm: A pareto-adaptive metaheuristic to multiobjective optimization. IEEE Trans. Syst. Man Cybernetics, Part A, 38(2), 258–277 (2008)
[25] Rudolph, G.: Self-adaptive mutations may lead to premature convergence. IEEE Trans. Evolutionary Computation, 5(4), 410–414 (2001)
[26] Back, T.: Generalized convergence models for tournament and (μ, λ)- selection. In Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc. 2–8 (1995)
[27] http://www.informatik.uni-freiburg.de/~stachnis/datasets.html