Robot Space Exploration Using Peano Paths Generated by Self-Organizing Maps

W.K. Lee[1], W. Duch[1,2], G.S. Ng[1]
[1]School of Computer Engineering, Nanyang Technological University
Nanyang Avenue, Singapore 639798
Department of Informatics, Nicolaus Copernicus University, Toruń, Poland

Abstract— **Autonomous exploration by a team of robots has many important applications in rescue operations, clearing of mine fields and other military applications, and even space exploration. With limited range of sensors robots have to divide exploration tasks among themselves working under multiple constraints. An optimal covering of two-dimensional area by robot trajectories requires formation of space-filling Peano curves. This may be achieved using Self-Organizing Feature Map (SOFM) algorithm. There are two steps involved in the proposed approach: first optimal trajectories are defined generating Peano curves for space of arbitrary shape using the SOFM algorithm, and second, robots are deployed for exploration based on selection of start/end nodes and radius of robot sensors. The same approach may be used to direct people or teams exploring some area in rescue operations. Tests simulations show that this approach achieves better coverage and faster exploration than competing algorithms.**

I. Introduction

In rescue operations teams of people, vehicles or autonomous robots have to cover some area performing search using sensors with a limited range. In case of search in mountain region the area may have a complex shape and quick optimal placement of searching teams may be critical. Clearing mine fields is usually done by military engineers, dogs or mechanical devices. Optimal trajectories have to be defined to cover the whole area without gaps. Autonomous robotic exploration of the unknown environments may also find applications in space exploration. For simplicity the term "robotic exploration" will be used, although the same algorithms may be used to optimize the human teams search process.

The problem has been widely studied from the perspective of a single mobile robot wandering through an unknown environment and surface, especially within the scope of simultaneous localization and mapping. However, when a team of multiple robots are deployed for exploration, the original problem is augmented with the requirement of coordinating the team of robots in order to complete the exploration task in a shorter period of time or, alternatively, being able to explore larger environments in a fair amount of time. In such a context, coordination strategies typically aim at maximizing the overall efficiency basically by minimizing repeated coverage of the workspace, while ensuring that all teams are kept busy exploring new areas without leaving any gaps.

The majority of approaches to multi-robot coordination for exploration purposes assume that the whole workspace is represented by a regular occupancy-grid (map) or nodes on the graph [1]-[8] in which cells basically correspond to obstacles, free space or an unknown space. The task can then be summarized as assessing whether the unknown cells correspond to obstacles or a free space. That map can either be centralized or distributed among the participating robots. In both cases, however, it is supposed that the absolute position of the robots with respect to the global map is known or can be easily determined. Exploring robots usually rely on some type of range sensors (such as laser, sonar, or stereo camera) which allow them to determine the presence or absence of obstacles within a specific maximum distance (sensor range), depending on the sensor's technology. In some cases though the sensor radius is supposed to be close to zero, and thus the main goal may be to ensure that the search team passes through all points of the free space, providing full space coverage.

Many algorithms to solve the exploration problem have been proposed [1]-[8], including sophisticated graph-search algorithms. The goal is to create trajectories for the exploration that allow to cover the whole area without any gaps in the shortest possible time. Balch and Arkin [9] compare the task of exploration to the grazing of an animal herd, distinguishing it from exploration, foraging, mapping and formation holding. These applications require the use of implicit information about the area and passive sensing, while other cooperative behaviors may require active communications. One of the most recent algorithms, the 'Repelling Forces' algorithm [9], has been inspired by physics. It is quite fast, allowing multiple robots to function individually and start their exploration from arbitrary points. It makes a rough search quite quickly, leaving numerous uncovered gaps, and it has a tendency to run into a deadlock because all the parameters between the walls and objects have fallen into equilibrium.

The main objective of this paper is to introduce, implement and test a novel algorithm for multiple robots space exploration, covering as much area in the shortest time possible. The general idea is to generate trajectories that fill the area with complex geometrical shape (including excluded zones in the middle) in an uniform way. Such trajectories are known as Peano curves (they have been discovered by Guiseppe Peano in 1890), and have been described by

1-4244-0342-1/06/$20.00 ©2006 IEEE

mathematicians working on fractals [11]. Although they can be generated in a systematic way for rectangular surfaces analytical formulas are not useful if the area has complex borders or additional constraints are required. The simplest way to generate Peano space filling curves is to use the principle of self-organization. In the next section the method is outlined, in the third section numerical experiments are described and some comparisons with other algorithms made, while the last section contains discussion and conclusions.

II. METHODOLOGY

The self-organized feature map algorithm has been shown to generate Peano curves [12]. Many other competitive learning algorithms, such as the growing grid [13], or growing cell structures [14], are also capable of self-organization. Starting from one-dimensional array of nodes, each specializing in analysis of some area of the space, these algorithms will lead to uniform density approximation (see however discussion on faithful representation in [15],[17]), with each node covering a Voronoi cell and adjacent nodes in the array pointing to the adjacent regions of the space. Fragments of such arrays may serve as trajectories for individual robots that will move along them. Theoretically, this should greatly improve the efficiency of the robot exploration, which, in turn, will greatly reduce the time needed to cover the whole indicated surface.

A. Improvements made in DemoGNG to generate Peano curves

Although competitive learning algorithms are quite easy to implement we have use an existing application, DemoGNG [18], because it has a large number of algorithms available for experimentation. Experiments described below were done with the SOFM algorithm, although other algorithms available in this package could be use as well.

The main requirement is to be able to generate random samples of points from the search area. A few basic shapes have been used for experiments (square, rectangle, circle and ring), but any surface with irregular shapes can also be used. The number of nodes in one dimensional array is inversely related to the assumed range of sensors (in most experiments below 30 nodes have been used).

Prior to running the SOFM algorithm several parameters must be set: first, initial and final neighborhood sizes σ_i, σ_f, shrunk according to $\sigma(t) = \sigma_i (\sigma_f/\sigma_i)^{t/t\max}$ formula, and initial and final learning rates ε_i and ε_f, shrunk according to analogical formula $\varepsilon(t) = \varepsilon_i (\varepsilon_f/\varepsilon_i)^{t/t\max}$, where t is the iteration number and $t_{\max}$ is the maximum number of iterations. The formula for the neighborhood changes is:

$$h(r, r_c, t, \varepsilon, \sigma) = \varepsilon(t) \exp\left(-\|r - r_c\|^2 / \sigma^2(t)\right)$$

These values are important in generating an optimal Peano curve as a path for the robots to move along during exploration, and they also determine the time taken by the algorithm to converge. Hence, optimal values for all these factors must be found before the deployment of the robots for exploration.

B. Factors Determining The Deployment of Robots

Upon forming the Peano curves the trajectory for exploration by the robots have been marked out. The starting and the ending points of each robot path must be defined before their deployment. This step is important as it ensures that the robots will move along without overlapping or repeated exploration of the same area. The radius of the sensors will determine and affect the overall efficiency of this approach. This radius should be approximately equal to the half of the distance between two adjacent nodes, and more precisely to the half of the maximum distance of any two (non-adjacent) nodes.

III. EXPERIMENTS

To illustrate the performance and efficiency of this approach, a Java-based application has been written. Together with improvements made in the DemoGNG application this show the feasibility of our approach. The number of iterations $t_{\max}$ has been fixed at 40000, sufficiently large for generation of good Peano curves for relatively small number of nodes. In Fig. 1 an example of a curve after a few iterations using rectangular area is shown. The green points show the values of two parameters associated with each node, initialized at random, but already topographically ordered (adjacent nodes have values that are close to each other in the 2D space). If the reduction of the neighborhood size is to quick Peano curves will not form and this type of configurations will be reached.

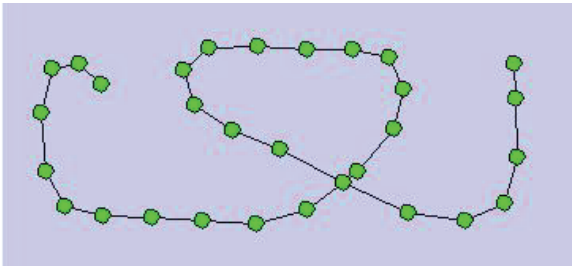

Fig. 1 Non-Peano curve obtained in step one.

A. Selection of optimal values of the neighborhood

To find the optimal values of σ_i and σ_f for the generation of the Peano Curve, the learning rates ε_i and ε_f after some experimentation were fixed at 0.05 and 0.005 respectively.

Using large initial value of σ_i =8.0 and σ_f =1.0 after convergence the nodes are in a line clustered together (Fig. 2). As should be expected, large neighborhoods prevent formation of Peano curves as all nodes are perturbed strongly by a good percentage of all the vectors.

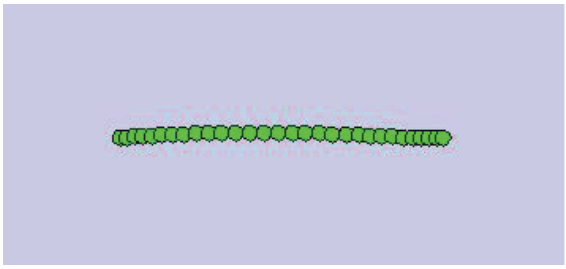

Fig. 2 Snapshot of nodes at σ_i=8.0 and σ_f=1.0.

Another extreme is to use all the time constant neighborhood, that is to set σ_i =1.0 and σ_f =1.0 (Fig. 3). The nodes have gradually started to form the Peano Curve. However, the algorithm has not converged completely in t_{max} =40000 iterations. Although in practical applications the time taken to execute this algorithm will hardly matter comparing with the actual robot exploration time and preparatory steps it is still worthwhile to search for better parameters.

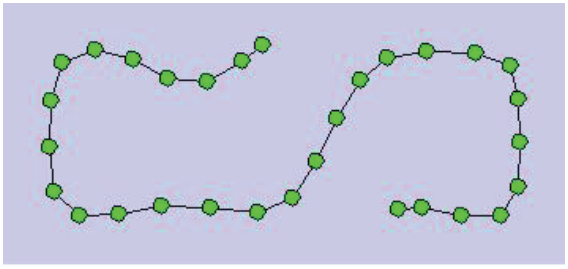

Fig. 3 **Snapshot of nodes at σ_i=1.0 and σ_f=1.0**.

The values of sigma determine the radius around the selected random signal which will influence and update the parameters of these corresponding nodes; the bigger the sigma values, the wider the radius of influence of the signals. After many tests the optimal values of σ_i have been determined in the range from 1.0 to 3.5, and of σ_f below 0.1 in order to obtain a curve resembling the Peano curve. Values of σ_i =3.5 and σ_f =0.01 are close to optimal, generating the curve show in Fig. 4.

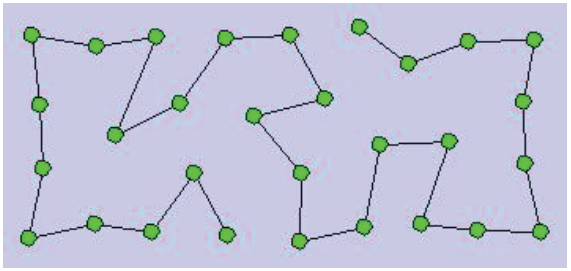

Fig. 4 **Snapshot of nodes at σ_i=3.5 and σ_f=0.01**.

B. Selection of optimal values of the learning rates

To find the optimal values of ε_i and ε_f for the generation of the Peano Curve, the values of σ_i =3.5 and σ_f=0.01 were fixed at their optimal values. Although this obviously does not guarantee that all four parameters will be optimal for this application it seems to be sufficient.

Using values of ε_i =1.0 and ε_f =1.0 (Fig. 5) the map obviously will never reach stability. Many edges are intersecting and there is no topographical neigborhood preservation. Furthermore, during the execution of the SOFM algorithm, the nodes are jumping around the surface. This is indeed a very undesirable case.

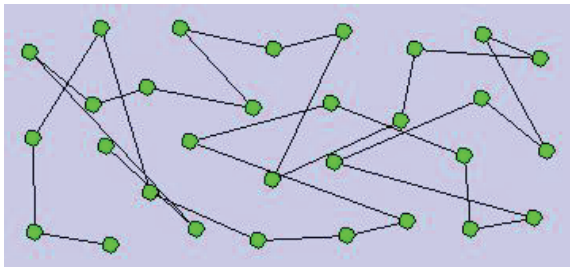

Fig. 5 **Snapshot of nodes at ε_i =1.0 and ε_f =1.0**.

Using values of ε_i =1.0 and ε_f =0.001 (Fig. 6): With the value initial value ε_i =1.0 much larger than the final value ε_i =0.001 the final curve converges to a rather decent Peano curve. As should be expected small ε_f values are needed to obtain the Peano paths.

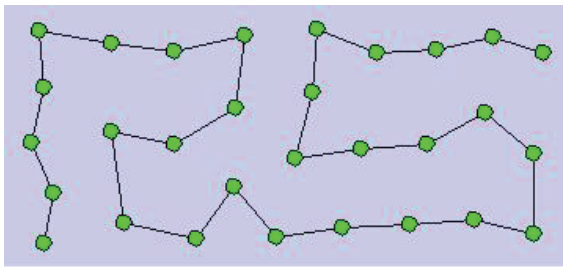

Fig. 6 **Snapshot of nodes at ε_i =1.0 and ε_f =0.001.**

After many experiments the values ε_i =0.05 and ε_f =0.0005 have been selected (Fig. 7), leading to stable and good looking mapping.

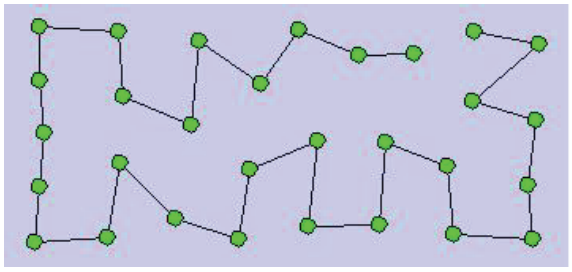

Fig. 7 **Snapshot of nodes at ε_i =0.05 and ε_f =0.0005.**

As seen from the above results and as should be expected from theoretical understanding of the process, the learning rates determine the stability of the learning process. It is possible to introduce numerical measures that will allow for comparison of the uniformity of the maps generated [15]-[17], but this will require a modification of the SOMF algorithm to achieve this; these possibilities are left for further work.

C. *Selection of Positions for Placement of Robots*

After execution of the SOFM algorithm and obtaining the Peano curve, placement of the robots on fragments of the Peano path should be considered. In the simplest case a uniform placement is advocated. For example, if a team of 5 robots are scheduled for a particular exploration exercise the best starting positions for these robots are at the node 1, node 7, node 13, node 19 and node 25, thus dividing the whole trajectory into 5 sections. Each robot should follow the route along the Peano path in the direction of the next node (linear interpolation), for example, the first robot should move from node 1 to node 2, node 3,…, and stop half way between node 6 and 7, covering the section assign to this trajectory.

D. *Selection of Values for Robot Sensor Radius*

Another factor that will influence the efficiency of the exploration is the range of the robot sensor values, that is to say, the radius of the robot sensor. The larger the radius of the sensor, the greater the area a robot can scan while moving along the Peano path. This means that efficiency should increase if the radius is large. However, the equipment used for the robot sensors may not be able to cover large areas. Hence, the gaps that are unexplored due to this factor may not be covered at all, as the robots are only deployed to move along the designated path, which is along the Peano path.

In the experiments showing the actual coverage of the robots exploring the territory the radius of the robot sensors is given in percentages of the maximum edge generated by the SOFM algorithm each time the proposed Peano trajectory is calculated. Of course in practice we should provide sufficient number of points along the trajectory to cover the whole area without gaps. This depends on the size and the shape of the area and therefore cannot be analytically computed. In the testing phase the number of robots is fixed at five and the number of trajectory points fixed at 30.

Assuming the robot sensor radius to reach 50% of the distance between the nodes (Fig. 8) leaves only a few uncovered small areas, therefore this radius is almost sufficient to cover most of the surface although a few more points should be used to improve it. To test this we can either increase the number of points or assume that the sensor range is slightly larger. These tests will help to calculate the number of nodes that should be used in generation of Peano curves.

Fig. 8 **Snapshot with robot radius at 50%.**

Using robot radius at 55% (Fig. 9): With an increase in the robot radius of 5%, there are no significant improvements in the case of the square and rectangle. However, the surface of the ring has more area covered while the circle is almost completely explored.

Fig. 9 **Snapshot with robot radius at 55%.**

Using robot radius at 60% (Fig. 10): Now, the results are definitely desirable. The whole surface of the square, rectangle and circle are more or less fully covered without significant overlaps. There are also considerable improvements in the case of the ring. Moreover, most areas can be subdivided into basic shapes such as square, rectangle and circle. Hence, robot radius at 60% can be taken as the optimal value for this size of the area.

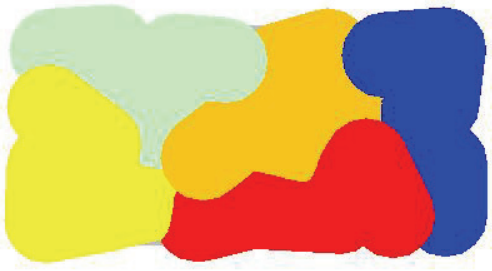

Fig. 10 **Snapshot with robot radius at 60%.**

Using robot radius at 70% and 75% (Fig. 11): For completeness the optimal value for full exploration of ring surface is radius at 70-75%.

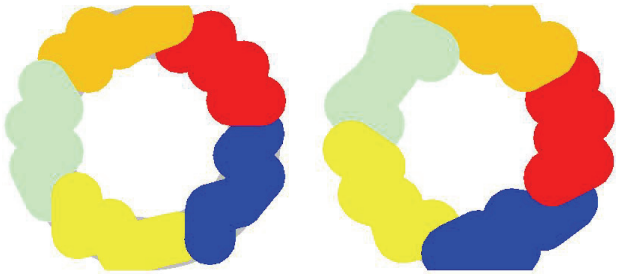

Fig. 11 **Snapshot with robot radius at 70% and 75% respectively**.

E. *Testing the efficiency of multi-robot exploration*

One of the basic factors which influences the total time and efficiency of the exploration process is the number of robots used in the exploration. Is the time for the exploration using

this algorithm linearly proportional to the number of robots? In the experiments below the robot sensor radius is assumed to be optimal for most shapes, that is about 60% of the maximum distances between the nodes.

Using a single robot, Table I gives time needed to move along the path (proportional to the length of the path that should be explored), and the total times taken for the whole exploration process, including the times for the generation of the trajectories. The total times are given for orientation purposes only, as in practice they may be optimized further. Upon comparison with the Collaborative MultiRobot Exploration application [10], it can be seen that this approach is definitely much faster with a total timing of about 60s in contrast with 270s for the square surface exploration, and the coverage is more uniform.

TABLE I
EFFICIENCY OF THE EXPLORATION WITH 1 ROBOT

Shape Of Surface	Time Taken For Robots To Move From Start Node To Destination Node (s)	Total Time Taken For The Whole Exploration Process (s)
Rectangle	26.8	66.8
Circle	17	57
Ring	25.6	65.6
Square	19.3	59.3

Using two robots (Table II) placed at the beginning of their trajectories the efficiency of the exploration process is approximately halved. For the square configuration this gives a total time of about 50s in comparison with the Collaborative MultiRobot Exploration approach with a timing of 200s.

TABLE II
EFFICIENCY OF THE EXPLORATION WITH 2 ROBOTS

Shape Of Surface	Time Taken For Robots To Move From Start Node To Destination Node (s)	Total Time Taken For The Whole Exploration Process (s)
Rectangle	12.7	52.7
Circle	8.9	48.9
Ring	12.7	52.7
Square	9.5	49.5

Using three robots (Table III): As the number of robots increased from two to three, the efficiency of the exploration step increases, becoming about one third of that for the single robot. Another phenomenon observed is that the values of the total time taken becoming more or less the same for all shapes. This simply means that the efficiency of the exploration process for various shapes is gradually becoming similar with the increase in the number of robots used. Comparatively the Collaborative MultiRobot Exploration approach [10] is about 4 times slower.

Using 5 robots (Table IV): As the number of robots grows the time for exploration becomes short and the total time for various shapes is dominated by the SOMF convergence, which takes about 40 seconds. From three to five robots deployed for exploration, the total times taken is halved again. With an average of 44 seconds, this proposed method of exploration is definitely more efficient than the Collaborative MultiRobot Exploration approach.

TABLE III
EFFICIENCY OF THE EXPLORATION WITH 3 ROBOTS

Shape Of Surface	Time Taken For Robots To Move From Start Node To Destination Node (s)	Total Time Taken For The Whole Exploration Process (s)
Rectangle	8.8	48.8
Circle	5.8	45.8
Ring	8.4	48.4
Square	6.3	46.3

TABLE IV
EFFICIENCY OF THE EXPLORATION WITH 5 ROBOTS

Shape Of Surface	Time Taken For Robots To Move From Start Node To Destination Node (s)	Total Time Taken For The Whole Exploration Process (s)
Rectangle	4.9	44.9
Circle	3.5	43.5
Ring	4.7	44.7
Square	3.8	43.8

A comparison of times done between the efficiencies of the current approach and the Collaborative MultiRobot Exploration [10] approach is presented in Table V. Note that the unknown environment surface is taken to be a square, as used in the Collaborative MultiRobot Exploration.

TABLE V
COMPARISON BETWEEN PROPOSED APPROACH AND COLLABORATIVE MULTIROBOT EXPLORATION APPROACH

Number Of Robots	Total Time Taken For Proposed Approach (s)	Total Time Taken For The Collaborative MultiRobot Exploration Approach (s)
1	59.3	270
2	49.5	200
3	46.3	180

The total time taken for the whole exploration for the current work is approximately one fifth of that with the Collaborative MultiRobot Exploration. Te actual exploration time is more important, as in practice the time to generate trajectories may be negligible in comparison to actual preparation and exploration times. The Peano curves approach has not only a short total time taken for full exploration but avoids overlapping paths of the robots, and thus will have a signifcant advantage.

IV. CONCLUSION

Optimization of stationary positions of base station (for example for cellular telephony) using self-organized approach may be done using two-dimensional maps, but exploration of the area using one-dimensional maps seems to be a novel approach. The proposed approach has many advantages: it is efficient, regardless of the number of robots used; it may be applied to areas with complex geometrical shapes (either because of natural obstacles or information collected earlier in

some sub-areas where search is no longer necessary), it does not leave any area uncovered, it is good for sparse exploration because it quickly reaches unexplored regions. It is also relatively easy to allow for real-time re-optimization in case of a robot failure. This approach will be of benefit in many practical applications, including military robot teams for mapping enemy terrain, mine-sweeping robots, space exploration or other related fields.

The Peano curve algorithm may not complete the exploration if the number of points on the trajectory and the radius of the robot sensors are too small, leaving some areas unexplored. However, this can easily be solved with the increase in the number of nodes used in the 2-D space, generating Peano curves with maximum distance between any two points that is smaller than half of the sensor range. Optimal number of points may be found running a few experiments.

In some applications robot exploration may start directly from points in the middle of the trajectory (dropping robots in the proper place of the area), but in others all robots will have to start from the same point or meet at some point to establish communication. This rendezvous problem – meeting of robots that start from unknown positions – has been studied extensively from both theoretical [19] as well as simulation perspectives [5].

Optimization of exploration that starts from the same point is an interesting and practical problem that has not been discussed using our algorithm. We may deal with it in a sub-optimal way, trying to reach the nearest point on the trajectory and covering on the way some area that will be crossed by other robots. After the robot reach their positions at the start of the scheduled trajectories it should be possible to change the excluded areas and modify the Peano curve, re-optimizing the trajectories. In most applications the gain will be relatively small, therefore we have not attempted this here.

Acknowledgement: WD is grateful for the support by the Polish Committee for Scientific Research, research grant 2005-2007.

REFERENCES

[1] D. Latimer IV, H. Choset, S. Srinivasa, A. Hurst, “Towards sensor based coverage with robot teams”, IEEE Int. Conf. on Robotics and Automation, 2002, pp.961-967.

[2] I. Rekletis, G. Dudek, E. Milios, “Probabilistic cooperative localization and mapping in practice”, IEEE Int. Conf. on Robotics and Automation, 2003, pp.1907-1912.

[3] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, H. Younes, “Coordination for multi-robot exploration and mapping”, 7th Nat. Conf. on Artificial Intelligence 2000, pp. 852-858.

[4] J. Borenstein, H. Everett, L. Feng, and D. Wehe, Mobile robot positioning: Sensors and techniques. Journal of Robotic Systems, Special Issue on Mobile Robots, 14(4):231-249, 1996.

[5] N. Roy, G. Dudek. Collaborative Exploration and Rendezvous: Algorithms, Performance Bounds and Observations. Autonomous Robots, 11(2): 117-136, 2001.

[6] W. Burgard, D. Fox, M. Moors, R. Simmons and S. Thrun, “Collaborative multi-robot exploration”, *IEEE Int. Conf. on Robotics and Automation,* pp. 476-481, 2000.

[7] W.W. Cohen, Adaptive mapping and navigation by teams of simple robots. Robotics and Autonomous Systems, (18):411-434, 1996.

[8] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, Robotic exploration as graph construction. IEEE Transactions on Robotics and Automation, 7(6):859-865, 1991.

[9] T. Balch and R.C. Arkin, “Behavior-based Formation Control for Multi-robot Teams”, IEEE Transactions on Robotics and Automation, 14(6), 926-939, 1998.

[10] M.C. Choo, “Collaborative MultiRobot Exploration”, Bachelor Thesis, SCE, Nanyang Technological University, 2005.

[11] B. Mandelbrot, “The Fractal Geometry of Nature”, W.H.Freeman and Co., NY, 1982.

[12] T. Kohonen, “Self-Organizing Maps”. Springer Verlag, Berlin, 1995.

[13] B. Fritzke, ”Growing Grid – a self-organizing network with constant neighborhood range and adaptation strength”. Neural Processing Letters 2(5), 9-13, 1995.

[14] B. Fritzke, ”Growing cell structures – a self-organizing network for unsupervised and supervised learning”, Neural Networks 7(9), 1441-1460, 1994.

[15] T. Villmann, R. Der, M. Herrmann, T. Martinetz., ”Topology Presevation in Self-organizing Feature Maps: General Definition and Efficient Measurement. IEEE Trans. on Neural Networks 8(2), 256-266, 1997.

[16] M.M. Van Hulle, ”Faithful Representations and topographic maps: From distortion- to information-based self-organization”, New York: Wiley 2000.

[17] M.M. Van Hulle, ”Self-organizing maps: theory, design, and application”, Tokyo: Kaibundo 2001.

[18] H.S. Loos, B. Fritzke, DemoGNG, available at: www.neuroinformatik.ruhr-uni-bochum.de/ ini/VDM/research/gsn/ DemoGNG/GNG.html

[19] S. Alpern, “The rendezvous search problem”. SIAM Journal of Control and Optimization, 33(3):673-683, 1995.