Exploration with Two Cooperating Mobile Robots

MOHAMMAD AL KHAWALDAH
SALVATORE LIVATINO
LILY MENG
School of Engineering and Technology
University of Hertfordshire,
College Lane, Hatfield, AL10 9AB
U.K
m.d.al-khawaldah@herts.ac.uk
S.Livatino@herts.ac.uk
l.1.meng@herts.ac.uk

Abstract: - In this paper a new exploration algorithm using two cooperating robots is introduced. The new technique is a combination of wall-following exploration algorithm and frontier-based exploration algorithm. Furthermore, robots sweep the line-of-sight between them continuously; if they can see each other then the area between them is assigned as free. The aim is to decrease the exploration time and energy consumption. The proposed algorithm is divided into two stages: Firstly, one of the robots follows (detects) the entire of the environment walls. And secondly, they employ frontier-based algorithm to complete exploring the remained unexplored areas. During these two stages, the robots sweep the line-of-sight between them in each step to maximize the exploration efficiency.

Key-Words: - Cooperating robots, Exploration, Line-of-Sight, Wall following, frontiers, Map building.

1 Introduction

The exploration and mapping of an unknown environment is an important issue in mobile robot research because of its real-world applications, like path planning, and planetary exploration. Good maps require accurate positioning systems. There many proposed techniques can deal with this problem effectively [1-4]. There are many advantages for using multi-robot systems. For example, a group of robots are capable to perform a single task faster than a single robot. Furthermore, merging of overlapping information, in multi-robot systems, can help compensate for sensor uncertainty. In addition, a group of cooperating robots can localize themselves more accurately, especially if they have different sensor capabilities. But when robots operate in teams there is the risk of interference among them. Finally, as the number of robots increases, longer paths may become necessary to avoid collisions with other members[5, 6]

One of the most important key reasons for deploying teams of robots instead of single robots is to increase the efficiency by decreasing the exploration time. As the number of robots exploring an environment increases, the importance of the coordination among their actions increases. The difficulty of coordination strongly depends on the knowledge of the robots. When the robots know their relative positions and share a map of the area they have explored, then an efficient coordination can be achieved by guiding the individual robots into different, non-overlapping areas of the environment. That can be easily done by assigning the robots to different exploration frontiers (transitions from explored free-space to unexplored areas). On the other hand, when the robots do not know their relative locations, then it is not obvious how to coordinate them effectively, since the robots do not share a common frame of reference or common map.

Large number of the published works in multi-robot exploration field is based on frontier cells e.g. [5-23]. A frontier cell is any free cell (not occluded) that has at least one explored neighboring cells. Each robot chooses a frontier cell, for example the nearest one as in [22], to be its target and start travelling towards it. It is expected that it gains information about the unexplored area when it arrives. The way in which the robot chooses its target is an important task that controls the exploration process. None of these published works has introduced the line-of-sight technique to increase the exploration efficiency. In addition, we thought that there is a possibility to find a bitter method to reduce the overlap between the robots.

The line-of-sight technique, in which the robots work in teams of two has been employed in some published works e.g. [24-26]. In this technique, each robot depends on its partner to correct its position estimate. During the exploration one of the robots (an "intelligent landmark") is fixed while the other one explores and localize itself depending on its fixed partner's position. And after that they exchange their roles. A different Line-of-Sight procedure is employed by Rekleitis *et al*. [27]. They proposed an algorithm for the complete coverage of free space. The environment is divided into cells, of different shapes and sizes, and a relatively complex procedure is used to explore each cell with a number of robots. None of the papers mentioned above [24-27] has employed the frontier-based exploration algorithm. In addition, they were not tested with different obstacles distributions or different obstacles numbers.

In this paper new exploration algorithm with two cooperating robots is proposed. The Algorithm is divided into two stages: Firstly, one of the robots follows (detect) the entire of the environment walls. During this stage the robots sweep the line-of-sight between them in each step. Secondly they employ the same frontier-based exploration algorithm introduced in [22], but with line-of-sight facility to complete exploring the remained unexplored areas. We tested our new exploration algorithms with different obstacle distributions and with different obstacles numbers. Eventually, we compared the results of our new exploration algorithm with the results of one of the close exploration algorithms published in the literature. The results show that our technique has new advantages over the existing techniques.

2 Wall-Follow Algorithm

This paper proposes a new algorithm for wall following to be employed when two cooperating robots are used to explore different environments. The algorithm is based on the principle that when the two robots are directed to frontier cells that keep them far away from each other (especially when each robot follows an environment wall), the exploration efficiency is larger (i.e. the energy consumed by the robots and the exploration time is less). The algorithm will functions in cooperation with new and relatively complex procedures employing the line-of-sight technique to increase the exploration efficiency. Our wall following algorithm directly guides the robots towards the environment walls to sweep (explore) as much cells as possible in each step. The approach we follow is an extension of the work in [24-27]with new improvements.

The proposed exploration algorithm can be summarized as follows:

1. Call the two robots A &B. A is known as the "wall follower" and B as the "trouble shooter". Both of them start at one of the environment corners or walls.
2. The wall follower starts following the walls. During each step of its movement it sweeps the line-of-sight to the other robot. It can also potentially correct its location estimate by using the trouble shooter as an intelligent land mark. It continues following the walls until the line-of-sight between the two robots is lost. Then it moves one back step to get the line-of-sight back again, and then it stops.
3. The trouble shooter starts moving toward the wall follower to discover the cause of the line-of-sight obstruction which would be either an obstacle or a wall. During this movement the line-of-sight would be available.
4. When the trouble shooter reaches the cause of the line-of-sight blockage, it starts following the walls in clockwise direction if the cause of obstruction is on its right hand side. On the other hand, if the cause of the obstruction is on its left hand side, it starts following the walls in counter-clockwise direction.

During the trouble shooter wall following, if it meets its partner (the cause of line-of-sight obstruction in this case should be a wall) then the procedure goes to point number 2. Wall following by robot B (the trouble shooter) continues until the line-of-sight is lost again.

5. The trouble shooter moves one back step and gets the line-of-sight again.
6. The procedure points (2-5) are repeated until the wall follower completes detection of all the environment walls.
7. The remained unexplored area is explored by using the same frontier-cells technique employed in [4] but combined with line-of-sight technique mentioned above. The robots' position estimates can also be corrected during this stage but it slows down the exploration to make the robots take turns to move.

In this algorithm, one robot moves at a time until the entire environment walls are detected. So, it can be considered as an energy saving exploration method.

3 EXPERIMENTAL WORK

Our experiments were conducted using Netlogo software [28]. It is a very powerful simulation that allows us to simulate an occupancy-grid-based environment exploration process with different numbers of agents (robots). It is a very flexible tool in which the environment is simulated as an m-by-n grid of cells and each cell has information stored in variables. This tool allows us to repeat an experiment many times and store the results in an Excel file for further processing. In the literature there are many published works based on Netlogo e.g. [29-31]. We started with a set of assumptions which can be relaxed later.

3.1 Key assumptions

Concerning robots, it is assumed that:

1. Each robot is equipped with a 360^o sensor that can detect the occupancy of all its eight neighbors. Also, it can distinguish between a robot and an obstacle when the other robot is in a neighbouring cell. This process is known as "scanning"
2. Each robot can see a ray from its partner, if there are no obstacles or other robots on the line between them (Line-of-sight technique).
3. Each robot knows exactly its own position and the position of its partner.
4. Robots move between the centers of cells.
5. All robots require equal time (a single tick in Netlogo) to perform a 360^o scan, try to make line-of-sight with its partner, and move to any neighboring cell.
6. Each robot can access a shared map which is continuously updated.
7. The communication between two partners is perfect.
8. Robots deal with environment edges as occupied cells.

3.2 Exploration Methodology

Each cell is allocated a state as shown in Table 1 below. Fig.1 below shows two robots when there is LOS between them. The red dots indicate obstacle positions. Fig.2 below shows two robots when there is NLOS between them. Fig.3 shows a completed map.

The exploration process runs as follows:

1. When a robot visits a cell, then all of its free neighbors are assigned to be "S" by scanning

Patch Code	Meaning	Patch Displayed Color
F	"Fresh" No Idea Yet	Gray
S	"Free" by Scanning	Brown
LOS	"Free" by Line-of-Sight Only	Red
S+LOS	"Free" by Scanning and Line-of- Sight	Yellow
NLOS	"Potentially Occupied" by No-Line-of-Sight	Blue
O	"Occupied" by Scanning	Orange

Table 1 Cells states and codes

2. If a robot scans a LOS cell, it is assigned to be "S+LOS".
3. If two robots in a team can see each other, then;
   3.1 All the F and NLOS cells on the line of sight between the two robots are assigned to be "LOS" cells.
   3.2 All the S cells on the line of sight between the two robots are assigned to be "S+LOS" cells.
4. If two robots in a team cannot see each other, then all the F cells between them are assigned to be "NLOS" cells.
5. If a robot scans a cell and discovers that there is an obstacle in that cell, this cell is assigned to be an "O" cell until the end of the exploration.
6. The exploration process is stopped when all the cells are explored (free or occupied).

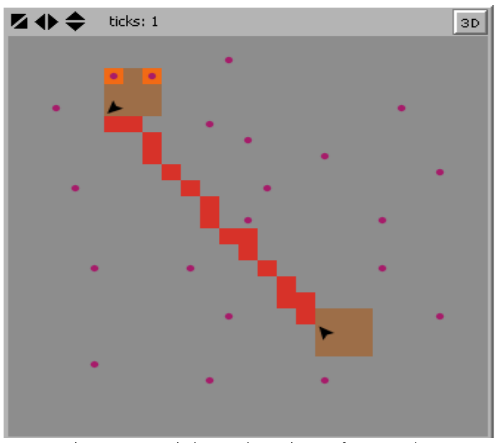


Fig.1: One tick exploration of two robots with LOS available between them

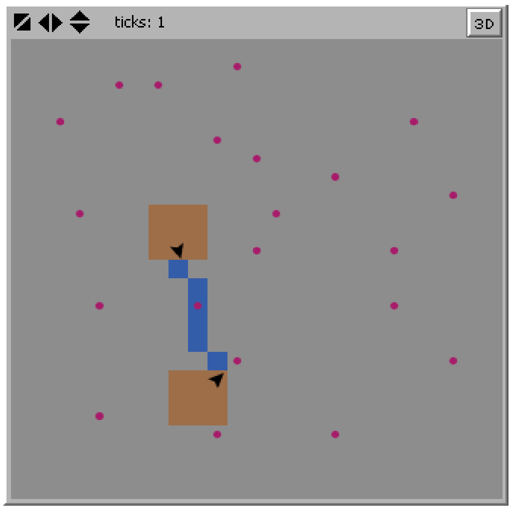


Figure 2: One tick exploration of two robots with NLOS between them

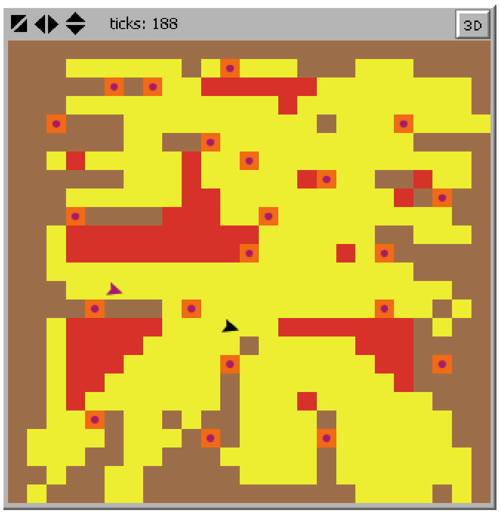
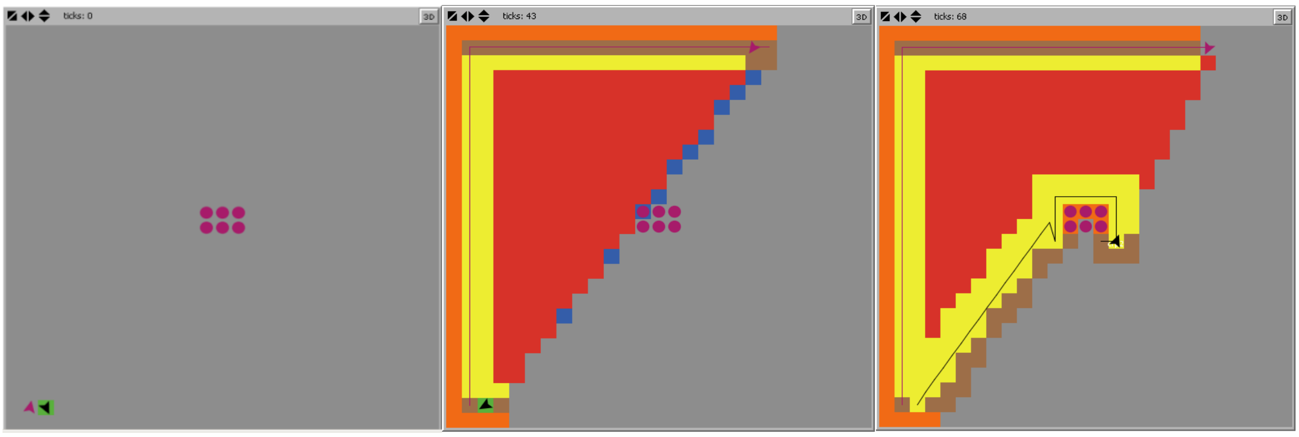


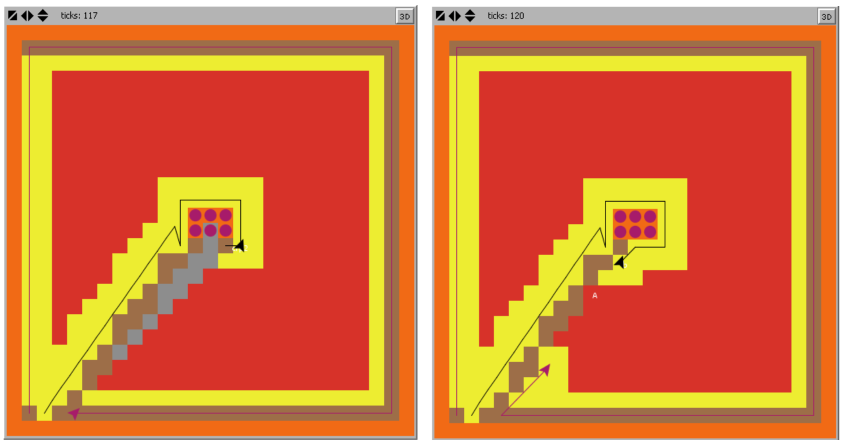
Figure 3: A completed map

Following are three exploration examples based on our wall-follow Algorithm detailed earlier.

1. Environment of one block of obstacles shown in figure 4



A B C

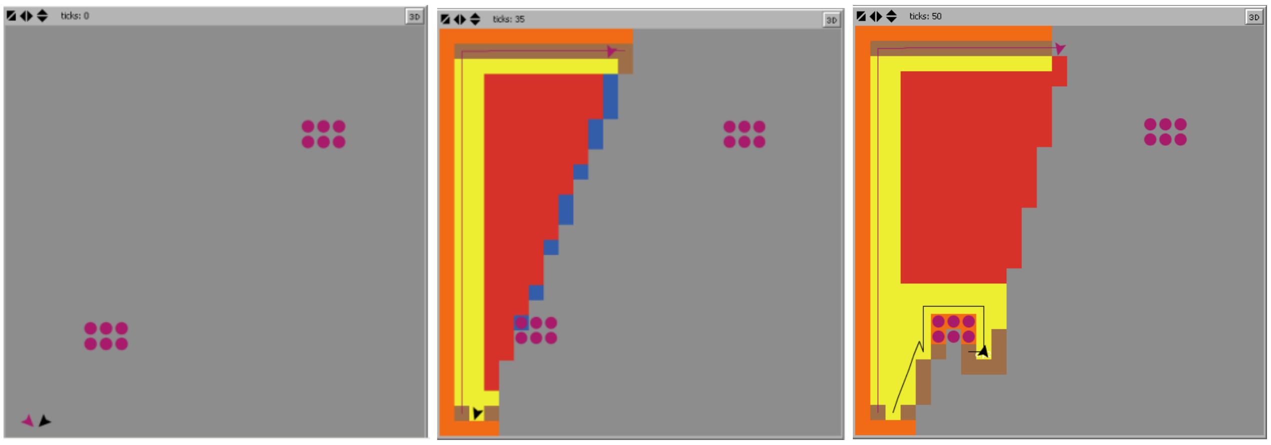


D E

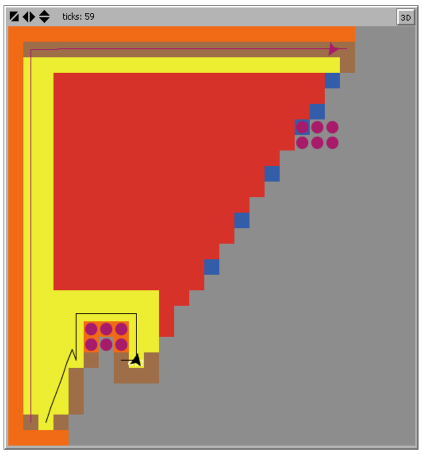
Figure 4: One block of obstacles. Environment exploration stages with Slow Wall-Follow algorithm

The robots start at an environment corner as shown in Figure 4A. The wall follower starts following the walls and sweeping the line-of-sight until the line-of-sight is lost, and then it goes back one step to get the line-of-sight again as shown in Figure 4B. The trouble shooter now starts moving towards the wall follower until it finds the cause for the line-of-sight obstruction, then it starts following the walls (around the block of the obstacles) and sweeping the line-of-sight at each step. The line-of-sight is lost again when the trouble shooter goes behind the block of obstacles. So, it goes one back step to get the line-of-sight again as shown in Figure 4C. The wall follower now continues following the walls until the walls of the environment are finished as shown in Figure 4D. The small remaining unexplored strip between the robots shown in Figure 4D is explored by the frontier-based employed in [22] but with line-of-sight facility, as shown in Figure 4E.

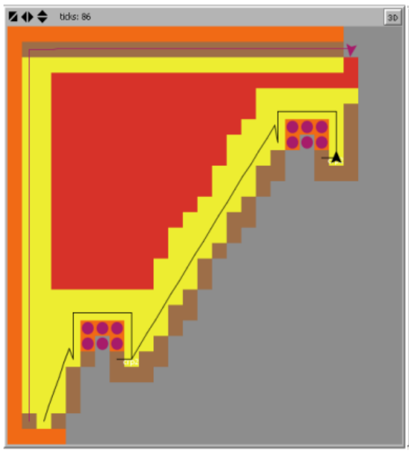
2. Environment of two blocks of obstacles shown in figure 5.



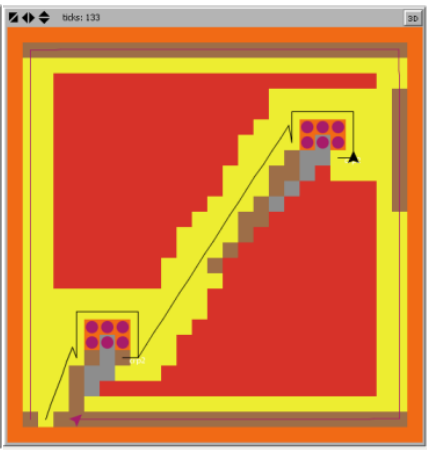
A B C



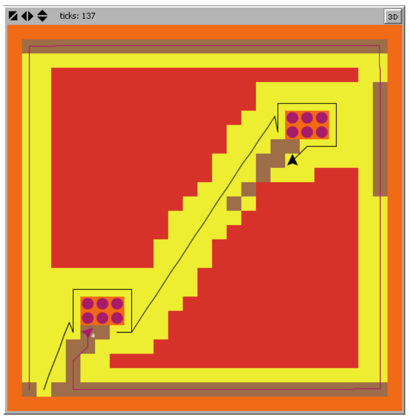
D



E



F



G

Figure 5: Two blocks of obstacles. Environment exploration stages with Slow Wall-Follow algorithm

The robots start at an environment corner as shown in Figure 5A. The wall follower starts following the walls until the line-of-sight is lost, and then it goes back one step to get the line-of-sight again as shown in Figure 5B. The trouble shooter now starts walking towards the wall follower until it finds the cause for the line-of-sight obstruction (the lower left block of obstacles), and then it starts following the walls (around the block of obstacles) and sweeping the line-of-sight at each step. The line-of-sight is lost again when the trouble shooter goes behind the lower left block of obstacles. So, it goes back one step to get the line-of-sight again as shown in Figure 5C. Now the wall follower continues following the walls again until the line-of-sight is lost (by the upper right block of obstacles) and then it goes back one step to get the line-of-sight again as shown in Figure 5D. The trouble shooter now starts walking towards the wall follower again until it find the cause for the line-of-sight obstruction (the upper right block of obstacles), and then it starts following the walls (around the upper right block of obstacles) and sweeping the line-of-sight at each step. The line-of-sight is lost again when the trouble shooter goes behind the upper right block of obstacles. So, it goes back one step to get the line-of-sight again as shown in Figure 5E. The wall follower now continues following the walls until the walls of the environment are finished as shown in Figure 5F. The small remaining unexplored two islands between the robots in Figure 5F are explored by the frontier-based mentioned in [22] but with line-of-sight facility as shown in Figure 5G.

3. Environment of three blocks of obstacles with a convex vertex shown in Fig.6

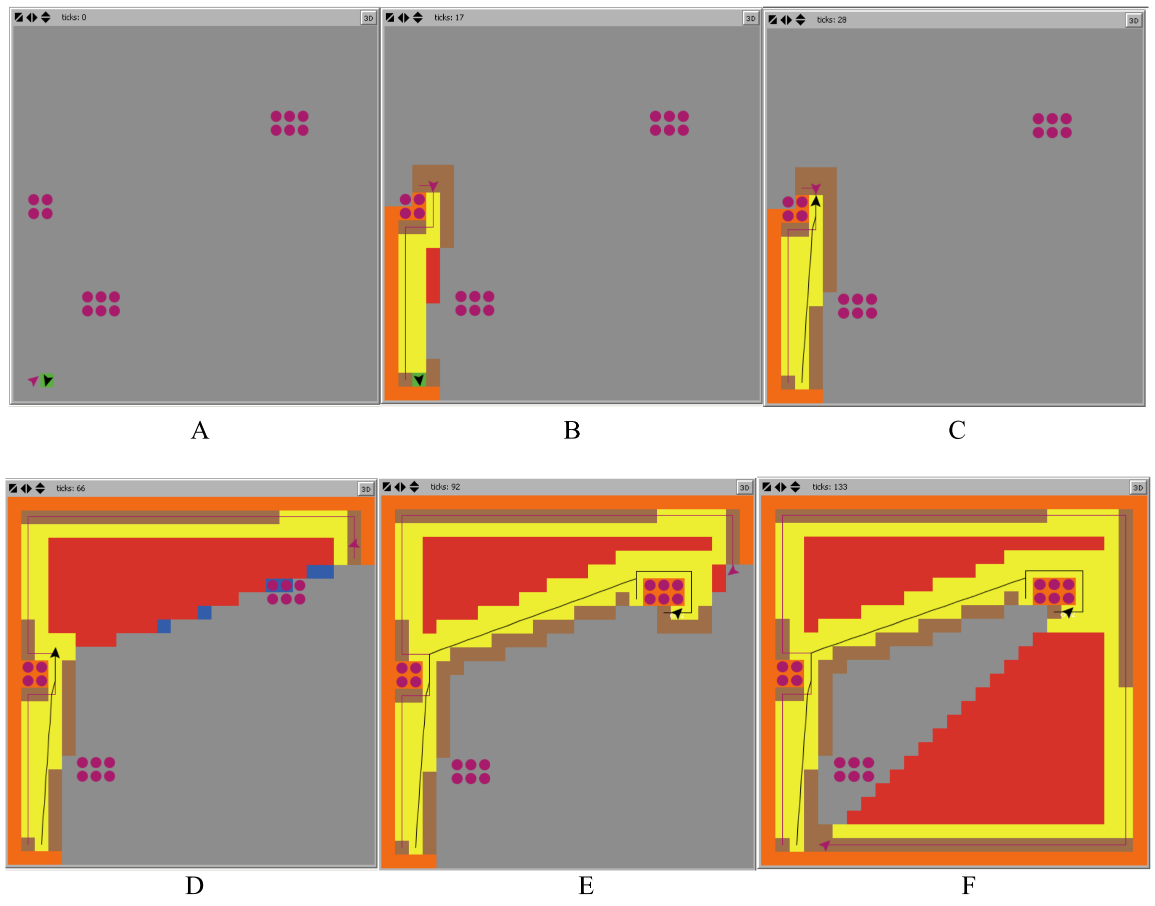


Fig.6 Exploration stages with Wall-Follow algorithm for an environment with three blocks of obstacles and containing a convex shape.

The two robots start at an environment corner as shown in Figure 6A. The wall follower starts following the walls until the line-of-sight is lost, and then it goes back one step to get the line-of-sight again as shown in Fig.6B. The trouble shooter now starts moving towards the wall follower until it finds the cause for the line-of-sight obstruction which is in this case the four obstacle block in middle left hand side of the environment. Because this block is on the trouble shooter left hand side, the trouble shooter follows the walls of this block in a counter-clockwise direction. While the trouble shooter wall following it meets the wall follower as shown in Fig.6C. Consequently, the wall follower continues following the walls again, and the trouble shooter waits until its turn comes, until the line-of-sight is lost (by the upper right block of obstacles) and then it goes back one step to get the line-of-sight again as shown in Fig.6D. The trouble shooter now starts moving towards the wall follower again until it finds the cause for the line-of-sight obstruction (the upper right block of obstacles), and then it starts following the walls (around the upper right block of the obstacles) and sweeping the line-of-sight at each step.

The line-of-sight is lost again when the trouble shooter goes behind the upper right block of obstacles. So, it goes back one step to get the line-of-sight again as shown in Fig.4E. The wall follower now continues following the walls and sweeping the line-of-sight until the walls of the environment are finished as shown in Fig.4F. The small remained unexplored island in Fig.4F is explored by the same frontier-based exploration algorithm detailed in [22]but with a line-of-sight facility as mentioned earlier.

It is clear that in our wall-follow algorithm the exploration time varies not only with the number of obstacles but also with the distribution of the obstacles themselves. And to investigate the effect of varying the obstacle distribution (positions of the obstacles) on the exploration time, a number of experiments have been repeated with the same number of obstacles but with different obstacle positions. Each experiment is repeated ten times except the experiments of the blocks of obstacles they are repeated just five times because they take

Experiment No.	One Obstacle	Two Scattered Obstacles	Five Scattered Obstacles	Ten Scattered Obstacles	One Block of Obstacles (Six Obstacles)	Two Blocks of Obstacles (Six Obstacles Each)
1	106	127	149	187	113	148
2	117	123	186	188	122	139
3	109	121	164	188	117	140
4	126	132	181	199	129	157
5	115	134	151	201	118	145
6	119	129	145	191	-----	-----
7	113	133	162	177	------	-----
8	113	129	137	197	------	------
9	118	146	172	181	------	------
10	110	134	179	191	-------	------
Average	114.6	130.8	162.6	190	119.8	145.8
S.Deviation	5.7	6.9	16.7	7.6	6.0	7.2

long setup time. Table 2 below shows the results of these experiments.

It is clear from Table 2 that changing the positions of the obstacles does not dramatically affect the exploration time. For instance, if all of the obstacles are close to each other, the trouble shooter will not need to travel long distances to find the causes of the line-of-sight obstruction. On the other hand, if the obstacles are far away from each other, the worst case would be if each corner has one or more obstacles near to it, then the trouble shooter will spend plenty of time detecting the causes for the line-of-sight obstructions. As a result, the exploration time would increase.

Table 2: Number of steps for repeated experiments with same number of obstacles and different obstacle positions with Slow wall-follow algorithm

4 Results Comaprisons

In this section, the algorithm presented in this paper is compared across a range of environments. All of the environments are 25-by-25 (625 cells). The exploration time required to explore an environment with this size is reasonable and therefore we can try our algorithm with a large number of runs. The results of the experiments are shown below in Table 3.

In Table 3 we also compare our work with the work in [22] in which the robots choose their next frontier target cell according to the equation:

$g_i=w_1I_i- w_2D_i+w_3\lambda_i$ ……………………(2)

where:

I_i: The information gain for the frontier cell *i* (the number of unexplored cells within the robot sensor range but, at the same time, not in the range of other robots or target cells for other robots)

D_i: the shortest travelling distance to the frontier cell *i*.

λ_i : is the nearness measure.

w_1, w_2, and w_3 are the weights for these three parameters and respectively.

The nearness measure is included in this equation to keep the robots close to each other to guarantee the communication amongst them. But in our simulation it is assumed that the robots are operating within their communication range, we just focus on the exploration algorithms. And the robots can share their maps in each step. Therefore, the nearness measure (λ_i) in eq. (2) is ignored, by setting w_3 to zero, when we compare the results of our technique (which is based on the eq. (1)) with this technique based on equation (2). w_1 and ω_1 are set to 1 as recommended in [22].

Table 3 shows comparisons between the experiments' results for the both exploration techniques presented in this paper. In addition to the elapsed number of steps taken by the exploration, the table also shows "Estimated Motor Energy" (E.M.E), which is the sum of the number of steps taken by each robot. This value is intended to give an approximation of the amount of energy expended during the exploration, and is higher than the elapsed time when both robots move simultaneously. Figs.7 and 8 show how the estimated motor energy and the exploration time vary with the number of scattered obstacles according to Table 3.

The following points can be observed from the results in Table 3:

1. Our proposed wall-follow algorithm is better than the exploration algorithm in [4] in terms of robot motor energy consumption (see Fig.5).
2. The proposed exploration algorithm consumes less exploration time in environments with no obstacles or with few obstacles (less than five)(see Fig.6).

	Sheng et al 2006		Wall-Follow	
	Steps	E.M.E	Steps	E.M.E
No Obstacles	158.7	317.4	93	93
One Obstacle	155.4	310.8	114.6	114.6
Two Obstacles	155.7	311.4	130.8	150.5
Five Obstacles	155.2	310.4	162.6	179.8
10 Obstacles	153.3	306.6	190.0	217.8
One Block of Obstacles	157.2	314.4	119.8	123.0
Two Blocks of Obstacles	159.4	318.8	145.8	155.0

Table 3 Comparisons between the experiments' results for the exploration techniques presented in this paper. All results are averages across 10 experiments for each technique in each environment.

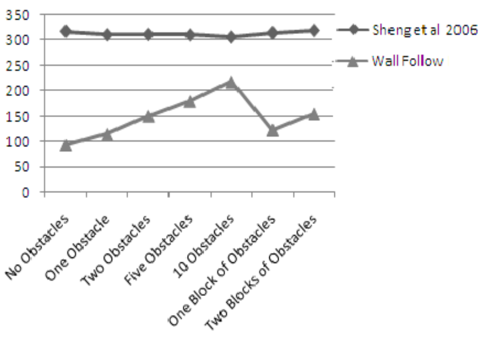


Fig.7 Estimated motor energy vs. Number of scattered obstacles

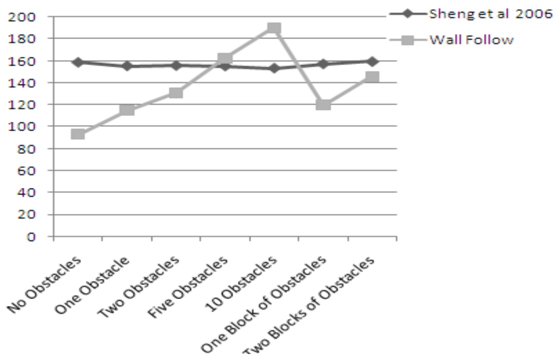


Fig.8 Exploration time (steps) vs. Number of scattered obstacles

5 Conclusions

In this paper a new exploration algorithm is proposed to explore an environment with two cooperating mobile robots. In this algorithm, one of the robots follows the environment walls and sweeps the free spaces in the environment as quick as possible by the line-of-sight technique. After that the robots switch to frontier-based with line-of sight exploration algorithm to explore the remaining unexplored part of the environment (if any). The results showed that the new proposed algorithm is very efficient to decrease the energy consumption by the robots. Furthermore, it decreases the exploration time for environments with no obstacles or with few obstacles (less than 5).

References:

1. Cumani, A. and A. Guiducci, *Fast point features for accurate visual odometry*. Proc. 12th WSEAS CSCC Multiconference, 2008.
2. Lorence, A.G., M.P.G. Gaffare, and J.A.S. de los Rيos. *Mobile robot global localization using just a visual landmark*. 2006: World Scientific and Engineering Academy and Society (WSEAS) Stevens Point, Wisconsin, USA.
3. Cumani, A. and A. Guiducci. *Mobile robot localisation with stereo vision*. 2005: World Scientific and Engineering Academy and Society (WSEAS) Stevens Point, Wisconsin, USA.
4. Chلvez, A. and H. Raposo. *Robot path planning using SIFT and sonar sensor fusion*. 2007: World

Scientific and Engineering Academy and Society (WSEAS) Stevens Point, Wisconsin, USA.

5. Burgard, W., et al., *Coordinated multi-robot exploration.* IEEE Transactions on Robotics, 2005. **21**(3): p. 376-386.
6. Alkhawaldah, M. and D. Lee, *Cooperative Robots Exploration with Line-of-Sight Technique* in *International Conference on Information and Communication Systems*. 2009: Jordan/Amman.
7. Scott, A.F. and Y. Changbin. *Cooperative multi-agent mapping and exploration in Webots&#x00AE*. in *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*. 2009.
8. Wurm, K.M., C. Stachniss, and W. Burgard. *Coordinated multi-robot exploration using a segmentation of the environment*. in *IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS* 2008.
9. Vazquez, J. and C. Malcolm. *Distributed multirobot exploration maintaining a mobile network*. in *Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference*. 2004.
10. Senthilkumar, K.S. and K.K. Bharadwaj. *An Efficient Global Optimization Approach to Multi Robot Path Exploration Problem Using Hybrid Genetic Algorithm*. in *ICIAFS. 4th International Conference on Information and Automation for Sustainability*. 2008.
11. Xin, M., Z. Qin, and L. Yibin. *Genetic Algorithm-based Multi-robot Cooperative Exploration*. in *IEEE International Conference on Control and Automation. ICCA* 2007.
12. Visser, A. and B.A. Slamet. *Including communication success in the estimation of information gain for multi-robot exploration*. in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, 2008. WiOPT 2008. 6th International Symposium on*. 2008.
13. Gifford, C.M., et al. *Low-cost multi-robot exploration and mapping*. in *IEEE International Conference on Technologies for Practical Robot Applications, TePRA* 2008.
14. Xin, M., et al. *Multi-agent-based Auctions for Multi-robot Exploration*. in *The Sixth World Congress on Intelligent Control and Automation. WCICA* 2006.
15. Hao, W., T. Guohui, and H. Bin. *Multi-robot collaboration exploration based on immune network model*. in *Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference on*. 2008.
16. Guangming, X., et al. *Multi-robot Exploration Based on Market Approach and Immune Optimizing Strategy*. in *Autonomic and Autonomous Systems, 2007. ICAS07. Third International Conference on*. 2007.
17. Poernomo, A.K. and Y. Huang Shell. *New Cost Function for Multi-Robot Exploration*. in *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*. 2006.
18. Zhao, J., X. Su, and J. Yan. *A Novel Strategy for Distributed Multi-robot Coordination in Area Exploration*. in *International Conference on Measuring Technology and Mechatronics Automation. ICMTMA* 2009.
19. Stachniss, C., O.M. Mozos, and W. Burgard. *Speeding-up multi-robot exploration by considering semantic place information*. in *Proceedings IEEE International Conference on Robotics and Automation, ICRA* . 2006.
20. Burgard, W., M. Moors, and F. Schneider, *Collaborative Exploration of Unknown Environments with Teams of Mobile Robots.* LECTURE NOTES IN COMPUTER SCIENCE, 2002: p. 52-70.
21. Rocha, R., J. Dias, and A. Carvalho, *Cooperative multi-robot systems: A study of vision-based 3-D mapping using information theory.* Robotics and Autonomous Systems, 2005. **53**(3-4): p. 282-311.
22. Sheng, W., et al., *Distributed multi-robot coordination in area exploration.* Robotics and Autonomous Systems, 2006. **54**(12): p. 945-955.
23. Fox, D., et al., *Distributed Multirobot Exploration and Mapping.* PROCEEDINGS-IEEE, 2006. **94**(7): p. 1325.
24. Rekleitis, I., G. Dudek, and E. Milios, *Multi-robot collaboration for robust exploration.* Annals of Mathematics and Artificial Intelligence, 2001. **31**(1): p. 7-40.
25. Rekleitis, I.M., G. Dudek, and E.E. Milios. *Multi-Robot Exploration of an Unknown Environment, Efficiently Reducing the Odometry Error.* in *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE.* 1997: LAWRENCE ERLBAUM ASSOCIATES LTD.
26. Rekleitis, I.M., G. Dudek, and E.E. Milios, *On multiagent exploration.* Visual Interface, 1998: p. 455-461.
27. Rekleitis, I., et al. *Limited Communication, Multi-Robot Team Based Coverage.* in *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*

2004: IEEE; 1999.

28. Wilensky, U., . 1999. NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.
29. Magg, S. and R.e.t. Boekhorst, *Pattern Formation in Homogeneous and Heterogeneous Swarms: Differences Between Versatile and Specialized Agents*, in *IEEE Symposium on Artificial Life*. 2007.
30. Niazi, M. and A. Hussain, *Agent-Based Tools for Modeling and Simulation of Self-Organization in Peer-to-Peer, Ad Hoc, and Other Complex Networks.* , in *IEEE Communications Magazine*. 2009.
31. Menezes, R. and H. Bullen. *A study of terrain coverage models.* in *ACM symposium on Applied computing.* . (2008). .