B.18

Utilizing Sensor-Based Robotic Terrain Coverage

Meral Camci and Sanem Sariel-Talay, *Member, IEEE*

Abstract—**This paper presents Enhanced - Least Recently Visited (ELRV) approach, an improved version of an existing sensor-based robotic terrain coverage algorithm, Least Recently Visited (LRV). LRV and ELRV are both proposed for either a single robot or a multi-robot team to cover an unstructured environment by incremental deployment of sensor nodes. LRV is used to make decisions on movement directions towards uncovered areas by means of the deployed sensors and their suggestions. Using sensors for coverage eliminates the need for mapping the environment. Unknown structure of the environment may sometimes result in inefficient coverage of the terrain as in the case of LRV. ELRV extends LRV by considering obstacle situations in unstructured environments. Furthermore, it suggests additional direction updates on sensor nodes to utilize coverage tasks. These supplementary direction updates are based on obstacle situations and decrease the total number of future direction updates when compared to LRV. Empirical evaluations of ELRV in the Player/Stage simulator present improved performance in terms of time to complete the coverage mission compared to LRV.**

Index Terms—**Sensor based coverage, dynamic coverage, robot sensor interaction, mobile robots.**

I. Introduction

EFFICIENCY in terrain coverage is an important concern for both robot applications (e.g., cleaning, mine-removal and search and rescue operations) and sensor-based environmental monitoring applications.

This study focuses on sensor-based robotic terrain coverage and proposes Enhanced - Least Recently Visited (ELRV) as an improvement on an existing algorithm, Least Recently Visited (LRV) [1]).

In LRV, robot deploys sensor nodes incrementally by using the information, on visited areas of the environment, provided by previously deployed nodes in an unknown environment. ELRV extends LRV by enhanced updating strategies for sensor nodes in initial deployment cases and for obstructed parts of the environment. These improvements help reducing overall time to complete the coverage mission.

II. Related Work

Choset [2] presents an extensive survey for coverage methods. Coverage algorithms are mainly classified into two groups namely online and offline coverage approaches [2]. In offline coverage [3], [4], [5], the map of the environment is available to the robot performing the coverage task. On the other hand, in online coverage [6], [1], [7], the map or the environment is not known by the robot.

M. Camci is with the Scientific and Technological Research Council of TURKEY (TUBITAK) Marmara Research Center, Information Technologies Institute, Gebze, Kocaeli, Turkey. (e-mail: meral.camci@bte.mam.gov.tr) M. Camci is also affiliated with Istanbul Technical University.

S. Sariel-Talay is with the Istanbul Technical University, Maslak, Istanbul, Turkey. (e-mail: sariel@itu.edu.tr).

Another important criterion considered for coverage algorithms [2] is the subject of the coverage task which is a robot itself or a robot interacting with sensors positioned (deployed) statically or deployed dynamically to the environment. In robot coverage [3], [4], [7], [5] the coverage mission is performed only by the robot and the information used in coverage is gained from only the robots' hardware (actuators and sensors of the robot). On the other hand, in sensor based coverage [6], [1] the robot interacts with sensors for completing coverage task. In sensor based coverage, sensors are either used in robotic coverage of an environment or they are directly used to form a network of sensors to monitor a terrain. Different deployment strategies may be performed for these applications. [6] uses static coverage in which sensors are positioned statically to cover every part of the environment with their sensor shadows at every time. This method is efficient for known environments (like offline coverage described in [2]). In LRV [1], as an online coverage strategy, a robot covers the environment step by step without having a map or plan. In this method the robot deploys sensor nodes incrementally according to the information obtained from the previous deployment status which is called dynamic coverage [1].

The number of robots performing the coverage task is another criterion in classifying the coverage methods. Coverage is performed by a single robot [1] or multiple robots [3], [6], [4], [5], [7]. Single robot coverage of an environment takes longer time when compared to multiple robot coverage. If robot failures occur, the coverage mission is performed by other robots in the team.

This paper focuses on an online strategy for a single robot to cover an unknown environment with the suggestions of the sensor nodes deployed incrementally. Particularly, LRV algorithm is investigated and some improvements are suggested as a new strategy on top of the main mechanisms of it. LRV has some deficiencies in deploying sensor nodes near obstacle locations and in some update strategies for newly deployed sensors. The main motivation of this research is to enhance LRV to eliminate these deficiencies.

III. Enhanced-LRV

LRV [1] makes use of sensor nodes for storing coverage information which is used by a robot to cover an unknown environment. Nodes are incrementally deployed by the robot based on some criteria and the movement directions of the robot are managed by the deployed nodes. During runtime, the robot deploys a new node whenever there is no node in its vicinity. This is the case for the initial deployment of the robot. There may be multiple nodes locating in its line-of-sight. In this case, it interacts with the node having highest signal strength. Throughout the paper, the *current node* represents

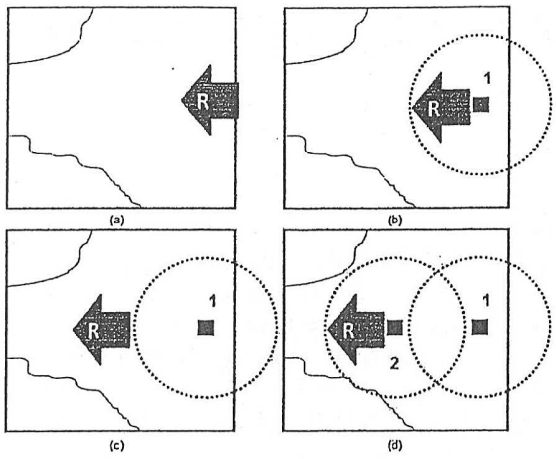

Fig. 1. A sample node deployment strategy by LRV. (a) The robot is initially deployed in the environment (b) It deploys a new node (c) It is out of the communication range with the current node (d) The robot deploys a new node

the node that the robot interacts with and the *closest node* represents the node with the highest signal strength after losing line of sight with the *current node*. Having the highest signal strength does not necessarily induce becoming the *current node*. As an example, a node having smaller signal strength may be selected as the *current node* since the candidate node (*closest node*) having the largest signal strength is occluded by obstacles. ELRV ensures this property by considering obstacle situations around the robot.

LRV algorithm is composed of three main parts. First part of the algorithm deals with the cases in which the robot is out of range of the *current node*. In this situation, the robot exploring/covering the environment deploys a new node. If there is a *closest node* in line-of-sight, it updates the weights of it by sending a message and moves according to this node's suggestions. In the meantime, this node becomes the *current node*. The second part of the algorithm is used for the cases in which the robot discovers obstacles on its way. According to the size of the obstacles, the robot either avoids the obstacle (if it is small) or it deploys a new node (if the obstacle is a large one and there are no other nodes except from the *current node* in the vicinity of robot) which then becomes the *current node*. The number of deployed nodes around large obstacles would be higher than that of obstacle-free areas due to this policy. This is a reasonable strategy to effectively cover the environment. The last part of the algorithm is used to forward the robot to the current direction if there are no obstacles on its way. During the traversal of the nodes, weight updates for directions are performed. These weights represent the number of traversals through the corresponding directions.

An illustration of the node deployment for LRV in obstacle-free areas is given in Fig. 1 (a-d). In Fig. 1(a), the robot is deployed in the environment and starts the coverage mission. In Fig. 1(b), the robot deploys the first node in the environment since there is not a *current node*. The node numbered with id 1 becomes the *current node* at this time, and the robot uses this information and decides on the next direction towards left side. Whenever the robot is out of communication range with

Algorithm 1 ELRV: Robot loop

robotDirection: direction which robot is going;
(n, d): (currentNode, suggestedDirection);
R: set containing data received from nodes in robot's vicinity (node id, suggested direction);
SHORT: communication range threshold used to determine when to deploy new nodes;
traversedDirection: direction which robot passes previously;
obsDirection: direction of obstacle when deploying node;

R = receive NODE_INFO messages from nodes in vicinity;
if out of SHORT communication range with **n then**
  ($n_{closest}$, $d_{closest}$) = node and corresponding direction in R with largest signal strength
  if $n! = NULL$ **then**
    Send(UPDATE_DIR, $n_{closest}$, $d_{closest}$)
    Send(UPDATE_DIR, $n_{closest}$, $traversedDirection$)
    if no obstacles detected in direction $d_{closest}$ **then**
      (n, d) =($n_{closest}$, $d_{closest}$)
    else
      Send(UPDATE_DIR, $n_{closest}$, $d_{closest}$)
      Wait for response, repeat the check
    end if
    $robotDirection = d_{closest}$
  else
    find an obstacle-free direction
    $robotDirection$ = obstacle-free direction
    deploy sensor node n' with suggested direction $robotDirection$
    (n, d) =(n' , $robotDirection$)
    Send(UPDATE_DIR, n, $robotDirection$)
    Send(UPDATE_DIR, n, $traversedDirection$)
    Detect obstructed direction(s)
    for each obstructed direction **do**
      Send(OBSTACLE_DIR, n, $obsDirection$)
    end for
  end if
end if
if moving and obstacle detected $\leq OBST\_AVOID\_RANGE$ **then**
  if obstacle is large and no nodes in vicinity **then**
    find an obstacle-free direction
    $robotDirection$ = obstacle-free direction
    deploy sensor node n' with suggested direction $robotDirection$
    (n, d) =(n' , $robotDirection$)
    Send(UPDATE_DIR, n, $robotDirection$)
    Send(UPDATE_DIR, n, $traversedDirection$)
    Detect obstructed direction(s)
    for each obstructed direction **do**
      Send(OBSTACLE_DIR, n, $obsDirection$)
    end for
  else
    avoid the obstacle
  end if
end if
if $robotDirection == NULL$ **then**
  Move in direction $robotDirection$
end if

the first node (refer to Fig. 1 (c)), it deploys the node 2 (refer to Fig. 1 (d)).

ELRV preserves the main parts of the LRV algorithm. It differs from LRV by its node deployment strategy in obstructed parts and its weight updating strategy.

ELRV enhances LRV in its node deployment strategy for

obstructed parts of the environment. The robot takes into consideration of the status of the environment during node deployment in ELRV. This strategy utilizes coverage in unstructured environments. The main procedures of ELRV are given in Algorithm 1.

In ELRV, weight updates for the directions are determined based on the obstacle status of the environment by the robot at the deployment time. The robot deploys a node to an obstacle-free direction (after avoiding the obstacle) and moves in that direction. There are two weight updates for the traversed and new directions that have been passed through by the robot when deploying that node. After deploying a node which becomes the *current node*, the robot sends two update messages to this node to declare that these two directions are traversed while deploying that node. When deploying that node, an obstacle-free direction is found. For that reason, there is no need to check if there is an obstacle in the deployed node's suggested direction and send messages to the related node for weight updates to find an obstacle-free direction in next steps (when the robot is in the range of the *closest node*) the robot is in the communication range of that node again.

In LRV, the node is deployed only considering the suggested direction information. In a new node deployment case, the traversed direction is not considered and the weight updates of these directions (suggested and traversed directions) during deployment are neglected. Also the suggested direction is not determined according to the obstacle situations. This result in additional direction updates to find an obstacle-free direction later when the robot is in the range of those related nodes.

Another enhancement made in ELRV during node deployment is updating the states of nodes according to the obstacle situation of the environment. ELRV updates the traversed and suggested directions as previously mentioned. The other directions except these traversed and suggested directions may be pointing to obstructed areas. In these situations, the robot sends related update messages for these obstructed directions to the deployed nodes. The messages are related to the obstructed-directions around the related deployed nodes. This strategy reduces the communication overhead and the total coverage time when compared to LRV. LRV ignores these obstacle situations and the robot can be directed to an obstructed direction by the corresponding node. This leads the robot to an untraversable direction and results in redundant efforts to avoid obstacles. In ELRV, after node deployment, the obstructed direction(s) are detected by the robot and the immediate update message(s) for these direction(s) are sent to the deployed node. The deployed node takes these update message(s) for the obstructed direction(s),and updates the weights of these directions with a high value in order to prevent suggesting and forwarding the robot to these directions.

In ELRV, the traversed direction of the *closest node* is updated instead of the opposite direction of the closest direction of the *closest node* as in the case of LRV.

Node loop update procedure of LRV is similar in ELRV except updating the obstructed direction at the time the node is deployed. This obstructed direction is updated with a high value for preventing the node to suggest this direction to robot.

In ELRV obstructed direction(s) and the traversed directions

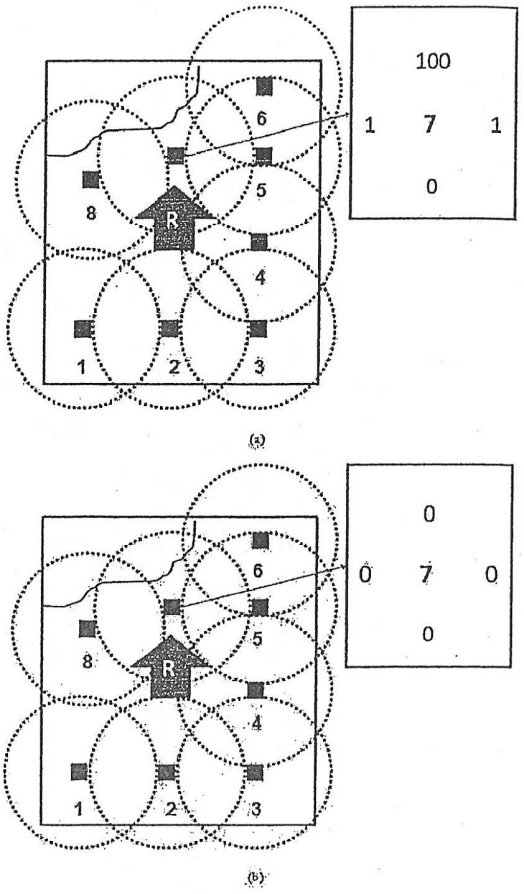

Fig. 2. Node deployment strategies (a) ELRV (b) LRV.

are updated in node deployment step. This results in immediate updates of the weights which are used later and reduces the communication overhead and the total coverage time. To clarify, a sample situation is illustrated in Fig. 2. Fig. 2 (a) outlines ELRV node deployment strategy. In this scenario, the robot starts exploring the environment from the left bottom corner and the nodes are deployed in counterclockwise order. The weight updates of the node 7, is given. Because the upper side of node 7 is obstructed, the corresponding direction is updated with high value for preventing that node to suggest this direction later. In this figure the traversed direction updates can also be seen. As opposed to LRV, ELRV updates the traversed directions at the time of deployment. Fig. 2 (b) illustrates the weight update strategy of LRV for the same scenario. In LRV, the obstructed direction is not considered and the next suggested direction according to the cross order rule [1] is selected as the upper direction. This yields additional direction updates to be performed later by the robot.

IV. Implementation Details

The Pioneer 2DX mobile robot equipped with 16 sonar range sensors positioned with 22.5 degrees interval is used in the simulations. Since the operation of the robot is managed by the sensors, obstacle detection and avoidance capability is considered for the robot. The sonar range finder sensors are

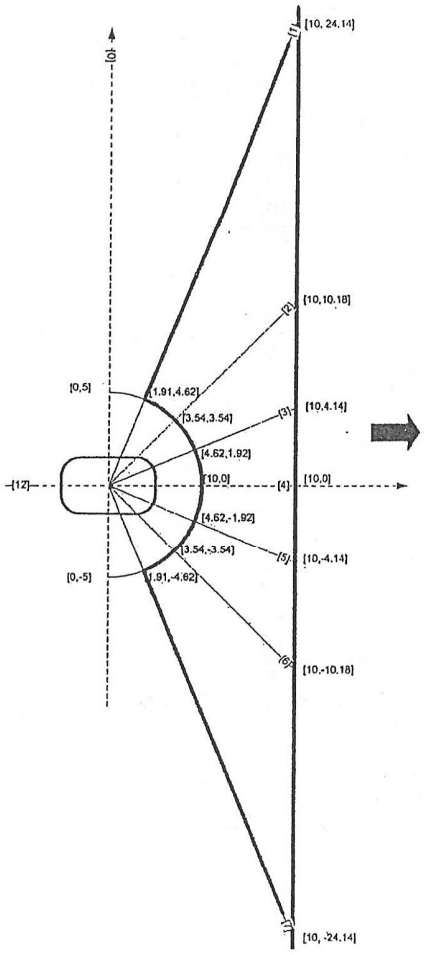

Fig. 3. Large obstacle detecting policy.

used for the robot to detect obstacles in its vicinity. In Fig. 3 the directions of the sonar range sensors are shown with the direction of the robot. The sonar readings are also used to determine the sizes of the obstacles to deploy new nodes in large obstacle situations. If all of these five sensor readings are in a range that is used for detecting large obstacles, according to algorithm LRV and ELRV a new node is deployed if there is no other nodes except the *current node* in the vicinity. All of the five sonar readings should be in the range drawn red in Fig. 3 to treat an obstacle as large.

The same obstacle avoidance and large obstacle detecting policies are applied in both algorithms to compare the results because there is no clear explanation of the obstacle avoidance policy in LRV. Also the same weight update strategy explained in [1] is used in both LRV and ELRV for the nodes managing robot to perform coverage mission.

V. Simulations and Experimental Results

ELRV is evaluated in Player/Stage [8], [9] simulator with a simulated Pioneer 2DX mobile robot having a sonar range finder.

Comparison criteria are selected as the percentage of the covered areas, time to complete the coverage mission and the number of nodes deployed. Total coverage result is obtained by calculating the number of uncovered points by the following intuitive formula:

$$Coverage\% = (Area - UncoveredArea)/Area \quad (1)$$

The simulations are ended when the total coverage is full for obstacle-free environments and minimum 98% coverage is obtained for environments with obstacles.

Experiments are designed in three sets for different environments with different obstacle densities. The performance of the algorithms are evaluated in empty environments, in 5 different environments with 10% obstacle densities and 20% obstacle densities in the first, second and the third set respectively.

The results of the first set of experiments are given as the average of 5 different runs in Table I.

The total coverage time, the coverage percentage and the number of nodes (given as a range) deployed are presented in the first, second and the third columns respectively. As the results illustrate, although the number of nodes deployed in the environment is similar in both LRV and ELRV, ELRV outperforms LRV in terms of time to complete the coverage mission.

The results of the second set of experiments are given as the averages of 5 different runs for each of the 5 different environments with 10% obstacle densities (Table II). The results of the third set of experiments are given as the averages of 5 different runs for each of the 5 different environments with 20% obstacle densities (Table III).

The presented results reveal that there is no significant difference in the number of deployed nodes comparing the two algorithms. The main difference of the results lies in the total coverage time of the environments which are either obstacle-free or obstructed. ELRV improves coverage efficiency approximately 33% in terms of time to cover the environment compared to LRV. This efficiency improvement is ensured by the node deployment strategy (obstructed directions update) and the traversed direction weight updates of nodes performed in ELRV. The new node deployment strategy considering the near obstacle information and updating weights accordingly prevents the robot from getting stuck into infinite loops. This strategy prevents a further forwarding to an obstacle location and accordingly extending the coverage time. According to the new node deployment strategy, additional direction updates for the traversed direction of the robot and the suggested direction of the node are performed. The robot uses the information gained when deploying a node without waiting any further steps.

It can also be observed from the experimental results that there is not a direct relation between the obstacle percentage of the environment and the coverage time.

TABLE I
Results of The Experiments in Obstacle-Free Environments

Method	Time (min,μ)	Coverage (%)	# of nodes (range)
ELRV	11.33	100	24-26
LRV	32.38	100	25-26

TABLE II
THE AVERAGE COVERAGE RESULTS FOR 5 DIFFERENT ENVIRONMENTS WITH 10% OBSTACLE DENSITIES

ENV %10	Time (min,μ)	Coverage (%)	# of nodes (range)
ELRV			
Env1	27.96	99.71	23-24
Env2	17.73	98.52	24-25
Env3	12.63	99.97	24-25
Env4	31.75	99.39	25
Env5	22.21	99.76	24-26
LRV			
Env1	89.58	98.23	25-26
Env2	50.68	99.53	24-26
Env3	34.33	98.64	25-26
Env4	93.75	99.14	26
Env5	67.93	99.25	24-25

TABLE III
THE AVERAGE COVERAGE RESULTS FOR 5 DIFFERENT ENVIRONMENTS WITH 20% OBSTACLE DENSITIES

ENV %20	Time (min, μ)	Coverage (%)	# of nodes (range)
ELRV			
Env6	11.88	99.36	23-25
Env7	26.50	99.11	25
Env8	21.35	98.62	24-25
Env9	30.20	99.71	26
Env10	28.61	99.34	25-26
LRV			
Env6	46.05	98.31	26
Env7	63.86	98.43	27
Env8	56.80	98.26	26-27
Env9	77.40	99.84	25-26
Env10	74.20	99.13	25

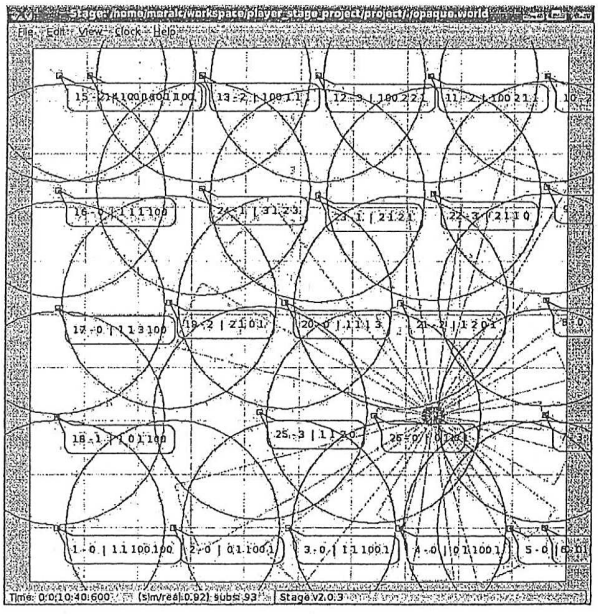

Fig. 4. Nodes deployed by ELRV in an obstacle-free environment.

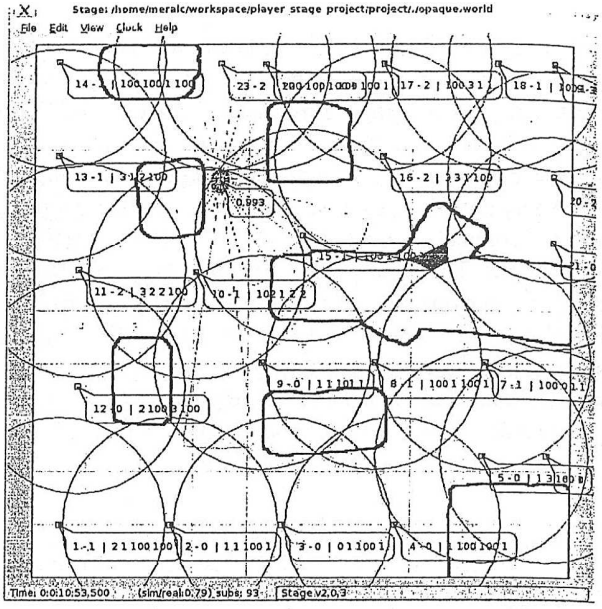

Fig. 5. Nodes deployed by ELRV in Env6 with 20% obstacle density.

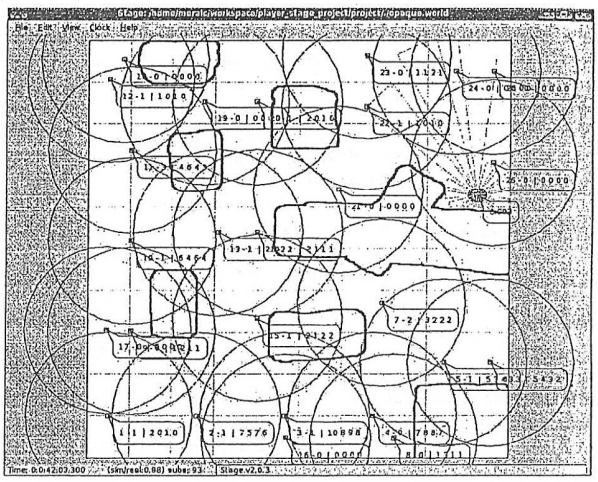

Fig. 6. Nodes deployed by LRV in Env6 with 20% obstacle density.

A snapshot of a simulation run for ELRV in an obstacle-free environment is given in Fig. 4. LRV performs nearly the same placement of nodes but with a longer coverage time as previously mentioned in Table I.

Fig. 5 and Fig. 6 illustrates sample snapshots from the simulation runs in an environment 20% obstacle density for the algorithms ELRV and LRV respectively.

A final remark illustrated in Fig. 5 is 100% coverage cannot always be achieved in obstructed environments because of the uncovered places remained in large obstacle regions (e.g., the uncovered region in Fig. 5 is painted with green in the figure).

VI. CONCLUSION

This paper presents ELRV as an enhanced extension of LRV. Unknown structure of the environment may sometimes result in inefficient coverage of the terrain as in the case of LRV. ELRV extends LRV by considering obstacle situations

and suggesting additional weight updates on sensor nodes to utilize the coverage mission. These supplementary updates ensure efficiency in coverage. Empirical evaluations of ELRV in the Player/Stage simulator present significant improvements provided by ELRV in reducing the coverage time compared to LRV. Analyzing the practical limitations of the proposed scheme and connectivity analysis of the nodes is left as a near-future work. The future work also includes integration of sensor failure recovery mechanisms to the design of ELRV.

Although the performance of ELRV is presented in a realistic simulator, porting ELRV on real robots and an extensive set of experiments for real environments are planned for analyzing the performance of ELRV to a greater extent.

REFERENCES

[1] M. A. Batalin and G. S. Sukhatme, "The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 661–675, 2007.

[2] H. Choset, "Coverage for robotics - a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[3] N. Agmon, N. Hazon, and G. A. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2006, pp. 1698–1703.

[4] G. A. Kaminka, "Redundancy, efficiency, and robustness in multi-robot coverage," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2005, pp. 735–741.

[5] X. Zheng and S. Koenig, "Robot coverage of terrain with non-uniform traversability," in *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2007, pp. 3757–3764.

[6] M. A. Batalin and G. S. Sukhatme, "Spreading out: A local approach to multi-robot coverage," in *Proc. Intl. Symp. Distributed Autonomous Robotic Systems (DARS)*, 2002, pp. 373–382.

[7] N. Hazon, F. Mieli, and G. A. Kaminka, "Towards robust on-line multi-robot coverage," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2006, pp. 1710–1715.

[8] B. P. Gerkey, R. Vaughan, K. Stoy, A. Howard, G. Sukhatme, and M. Mataric, "Most valuable player: A robot device server for distributed control," in *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2001, pp. 1226–1231.

[9] B. P. Gerkey, R. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. Int. Conf. on Advanced Robotics (ICAR)*, 2003, pp. 317–323.