Cooperative multi-agent mapping and exploration in Webots[®]

Adele F. Scott and Changbin Yu

*Research School of Information Sciences and Engineering, The Australian National University,
Canberra ACT 0200, Australia*

{u2566368, Brad.Yu}@anu.edu.au

Abstract—This paper addresses the problem of mapping and exploration of an unknown space by cooperative multi-agent systems. The exploration problem is extended to jointly covering an area n times. Agents are localised and can communicate with other agents. A probability based mapping algorithm is developed. Based upon this, a potential field based exploration algorithm is proposed, with three sample charge profiles that can be used for different mission requirements. Simulation results in Webots confirm the scalability and the effectiveness of these algorithms. This research justifies and prepares for a full trial in a multi-robot testbed.

I. INTRODUCTION

The problem of mapping and exploration of an unknown space by an autonomous agent has attracted much attention. The task becomes more challenging and interesting with multi-agent cooperative systems [1] [2] [3]. Solutions to such problems are of practical significance to applications such as military surveillance, bush fire monitoring, and mining exploration.

Multi-agent systems can be roughly categorised into centralised systems and decentralised systems. The former often assumes existence of a global controller, or master agent, that monitors and instructs each individual agent. The latter, which has become more and more popular, has no global controller. Individual agents are fully autonomous in motion planning. This however does not mean that they do not cooperate with each other. Cooperation is incorporated through the agents' onboard communication and sensing capabilities. Constraints on these capabilities are application specific. In many studies, it is assumed that all agents are localised and can communicate with all peer agents [4] [5], and will be assumed in this paper.

Several authors define the problem of multi-robot exploration as jointly entirely covering an area while minimising overlap between areas covered by each robot [4] [6], whereas [7] defines it as “the act of moving through an unknown environment while building a map that can be used for subsequent navigation”. [8] notes that some amount of repeated coverage may be desirable in a dynamic environment. These varying definitions emphasise the fact that exploration is an application specific task, and that different mission goals may call for different behaviours.

An extension of the exploration problem is proposed that includes efficiently jointly covering the map n times, with application specific priorities on the manner in which it is repeatedly covered. It may be desirable in some applications

for robots to simply jointly cover the area n times as quickly as possible without regard for the way in which they cover it. Such an application may be vacuuming, where it is perfectly acceptable to repeatedly cover a small area n times before another area has even been visited once. On the other hand, it may be important that the robots cover an area evenly and complete each coverage before beginning the next. Such an application may be surveillance, where it is not as useful if an area is re-covered many times in rapid succession rather than covering the entire area once before coming back for another sweep.

In order to solve this problem, an exploration algorithm that uses a 2D, grid-based, probabilistic map and a potential field model is proposed. Potential field based approaches have been suggested before [4] [9], which base the attraction of cells on the belief of whether they are an object or not, while in this paper, the magnitude of the attraction of cells is based on the certainty information stored in the proposed map. Certainty is increased with each passing of the robot, and so this approach provides a structured way in which to explore an environment n times. Three charge strategies are proposed that emphasise different behaviours intended to match the priorities described in the previous paragraph. As no prior map information is available, the path planning approach presented here is closed-loop and time critical so simplicity is of importance. A potential field approach provides this simplicity.

These algorithms can be verified at various levels, from simple Matlab simulations, to solutions created with robotics simulators, to actual experiments on a testbed. In this work, the solutions are fully tested with a simulation environment that integrates the Webots robotics simulator and Matlab programs. Webots has become a popular platform for simulating cooperative problems such as navigation [10], formation control [11] and task allocation [12]. Simulation in Webots enables immediate prototyping in a multi-robot testbed (currently under construction). Moreover, it facilitates the investigation of scalability through hardware-software co-simulation, where a number of real robots can cooperate with virtual robots in a unified environment.

Section II introduces the simulation environment. Mapping and exploration algorithms are explained in detail in Section III. Simulation results are presented and discussed in Section IV. Section V concludes the work.

Fig. 1. (a) The e-puck robot and (b) its Webots model.

II. SIMULATION ENVIRONMENT

Webots is a professional mobile robotics simulation package distributed by Cyberbotics [13]. It provides 3D visualisation of simulation environments and utilises the Open Dynamics Engine (ODE) for rigid body dynamics and collision detection. Physics can be extended with a custom plug-in to allow simulation of environments such as air or water. Custom robots can be created using a library of geometries, sensors and actuators, or one of the supplied robot models can be used. Each robot is run by a “controller”: an executable file compiled from a C, C++ or Java file. A special robot class, the supervisor, provides omnipotent functions such as changing the environment, sending messages to robots or recording trajectories. Robot position information was provided to robots in this paper using the supervisor. Webots supports transfer of controllers to many real robots, including the e-puck, Khepera and Aibo robots and others, in the form of cross-compilation and remote control. The remote control option allows the possibility of running several robots both in simulation and reality, providing a highly scalable yet realistic test environment.

For simulations in this paper, Webots was integrated with the Matlab engine [14] via the Matlab C API [15]. This interface provides a library of C functions that allow a C program to launch an instance of the Matlab engine, pass and extract variables to and from its workspace and execute commands. Matlab was used primarily as a back-end to process the map data, in order to take advantage of its matrix processing strength. Calculation of map updates, updating of the map and path planning were all conducted using Matlab. The remaining functions were performed using C in the robot controllers.

Webots provides models of many educational and research robots, including the e-puck robot used in this paper. The e-puck robot and the Webots model are shown in Fig. 1. The e-puck has a radius of 3.5cm and is driven by two independent stepper motors. It has eight infrared distance sensors with a range of about 6cm, which are of primary interest in this paper. The Webots model incorporated these sensors and models each using the profile described in Fig. 2. Additionally, the e-puck contains a Bluetooth chip to allow robot-robot communication. Communication was modelled with ideal transmitters and receivers in Webots; communication radii were considered infinite and transmission was noiseless.

The simulation environment described in this section forms the basis for analysis of the mapping and exploration algo-

Fig. 2. Distance sensor profile for the e-puck model showing relationship between raw sensor readings and actual distances. Noise is uniformly distributed within the error bars. The horizontal line corresponds with the upper noise bound at 0.07m and shows the threshold below which the sensor value is interpreted as free space.

rithms presented in Section III.

III. ALGORITHMS

The map presented here is a 2D, cell-based, probabilistic map created using distance sensor measurements. The environment is partitioned into equal sized *cells* that contain information on whether that cell is an *object* or *free space*, and also how certain the robot is of this belief. This information is recorded as a number in a user-defined range, where the upper half corresponds to free space, and the lower half corresponds to an object. Numbers further from the midpoint reflect a higher certainty of the classification of that cell.

The basis of the exploration algorithm is a potential field model of the world. The robots and map cells are modelled as charged particles with opposing polarity. Cells and other robots exert force in an inverse squared manner like Coulombic forces in an electrostatic field. This draws robots towards locally unexplored areas while encouraging them to spread out. The magnitude of the charge of each cell is determined by the value of the corresponding map cell. The exploration algorithm does not distinguish between cells recorded as objects and free space and hence does not recognise if an area cannot be explored, such as an area inside an object. This means that it is most suited to exploring open spaces.

Fig. 3 shows the overall closed-loop mapping and exploration approach. These steps are described in detail below.

Before proceeding with detailed descriptions of these steps, let us note the following global assumptions: the problem of localisation is not considered here and hence it is assumed that all robots have knowledge of their absolute position within the environment. This is easily accommodated in a testbed with a calibrated overhead camera, and in actual deployment (of a necessarily bigger agent than an e-puck), with GPS. In addition, all-to-all communication is assumed.

A. Mapping

The 2D world coordinate system is denoted by X and Z and is shown in Fig. 4. There are two important parameters involved in the map creation. They are the *map dimensions*, the range of X and Z and the size of each cell given in metres, and the *map range*, the range of values that each cell of the map can take on.

Fig. 3. A summary of the closed-loop multi-agent mapping and exploration approach.

 Fig. 4. Diagram of the robot and world reference frames, oxz and OXZ respectively.

1) *Initialise map and n* : The map is a 2D matrix sized according to the map dimensions with all cells initialised to the midpoint of the map range. The variable n provides a count of how many times the map has been entirely covered and is initialised to zero. Its purpose is described in the *Exploration* subsection later.

2) *Collect distance measurements*: Raw distance sensor measurements are read and converted into real distances, d , using the profile for the particular sensor used. The distance sensor profile for the e-puck model was shown in Fig. 2. Sensor measurements below the noise threshold shown were interpreted as free space, and d set to 0.06m, otherwise measurements were interpreted as an object d metres from the sensor.

3) *Calculate map updates*: Fig. 4 shows the relationship between a robot's reference frame and that of the world. While this is drawn with an e-puck, there is no loss of generality in this approach. The coordinates of the observation, P (either an object, or the edge of the distance sensor's vision), are calculated in the robot's reference frame via

$$P_{(oxz)} = d \begin{pmatrix} \cos \theta \\ -\sin \theta \end{pmatrix} + S_{(oxz)} \quad (1)$$

where d is the distance measurement previously found and $S_{(oxz)}$ and θ are the coordinates and orientation of the sensor with respect to the robot. Subscripts denote the coordinate system. P is transformed to the world coordinate system using the Euclidean transformation

$$P_{(OXZ)} = \begin{pmatrix} \cos \Theta & \sin \Theta \\ -\sin \Theta & \cos \Theta \end{pmatrix} P_{(oxz)} + o_{(OXZ)} \quad (2)$$

and similarly for S , where Θ is the orientation of the robot with respect to the world reference frame and o is the position of the robot.

The indices of the map corresponding to P are identified and noted as an object or free space according to the classification when distance measurements were made. The indices of cells in a straight line between the cells corresponding to S and P are also identified and noted as free space.

4) *Update map*: For every cell that was identified in the previous step, its value is incremented if it was classified as free space, and decremented if it was classified as an object.

Values of cells corresponding to the footprint of the robot are also incremented. These are cells that have coordinates (X, Z) which satisfy

$$(X - o_X)^2 + (Z - o_Z)^2 < r \quad (3)$$

where $(o_X, o_Z) = o_{(OXZ)}$ is the centre of the robot in the world coordinate system and r is the radius of the robot as in Fig. 4. Map values are constrained so as to not exceed the map range.

5) *Map merging*: The remaining steps up to path planning denote the sharing and merging of the map. As all robots have knowledge of their absolute position within the environment, they simply communicate their map update index pairs and classifications as well as their position, which peer robots can apply in exactly the same way as they applied their own map updates.

B. Exploration

This section describes the remaining two steps in Fig. 3: path planning and path execution.

1) *Path planning*: The goal of the path planning algorithm is to calculate a target angle based on the current state of the map, the robot's own position and the positions of peer robots. It is useful to interpret the map as a direct measure of how many times a cell has been visited. There are two parameters involved in calculating the target angle: the *robot charge*, the charge of each robot, and the *charge profile* of cells, a charge function dependent on the number of times a cell has been visited. Three different charge profiles, *dynamic-step (DS)*, *fixed-step (FS)* and *ramp (R)*, are presented.

First, the counter n is incremented if 95% of the map has been visited $n + 1$ times. Note that this percentage can be adjusted to suit any particular application.

The values of the map are converted into the number of times a cell has been visited using

$$V = |M - M_0| \quad (4)$$

where M is the map matrix, M_0 is the initial state of the map and V is the number of times each cell has been visited. The charge of each cell, is calculated using the selected charge

Fig. 5. The charge profiles used for the three potential field path planning strategies: (a) dynamic-step, (b) fixed-step and (c) ramp. n is the number of times the map has been entirely covered.

profile. The DS profile is given by

$$q_i(V_i, n) = \begin{cases} C & \text{if } 0 \leq V_i \leq n \\ 0 & \text{if } n < V_i \leq V_{\max} \end{cases}, \quad (5)$$

the FS profile is given by

$$q_i(V_i) = \begin{cases} C & \text{if } 0 \leq V_i < V_{\max} \\ 0 & \text{if } V_i = V_{\max} \end{cases}, \quad (6)$$

and the R profile is given by

$$q_i(V_i, n) = \begin{cases} C & \text{if } 0 \leq V_i \leq n \\ \frac{C}{V_{\max} - n}(V_{\max} - V_i) & \text{if } n < V_i \leq V_{\max} \end{cases} \quad (7)$$

where C is the maximum charge, $V_{\max}$ is the maximum number of visits as defined by the map range, V_i is the number of times the i th cell has been visited and q_i is the charge of the i th cell. Fig. 5 displays these three profiles. The total electrostatic force is given by

$$\mathbf{F} = Q_R \left(\sum_{i=1}^L \frac{q_i}{\|\mathbf{r}_i\|^2} \hat{\mathbf{r}}_i + \sum_{i=1}^{K-1} \frac{Q_R}{\|\mathbf{r}_i\|^2} \hat{\mathbf{r}}_i \right) \quad (8)$$

where L is the number of cells in the map, K is the number of robots, Q_R is the robot charge, and $\mathbf{r}_i$ is the vector from the i th cell or i th robot to the current robot. The target angle is the angle of this vector, given by

$$\varphi = \tan^{-1} \left(\frac{-F_z}{F_x} \right). \quad (9)$$

2) *Path execution*: The goal of the path execution algorithm is to steer the robot towards the target angle calculated by the path planning algorithm, and to avoid objects.

The difference between the target angle and the orientation of the robot is calculated, being

$$\Delta\varphi = \varphi - \left(\Theta + \frac{\pi}{2} \right). \quad (10)$$

Note that the extra $\frac{\pi}{2}$ is due to the definition of the coordinate system, see Fig. 4. If $-2\pi < \Delta\varphi < -\pi$ or $0 < \Delta\varphi < \pi$ the robot veers left, otherwise the robot veers right. Robots have only one set of wheel speeds for each direction. When turning left, the right wheel speed is set to maximum speed and the left to half this, and vice versa for turning right. This enforces a non-zero turning radius constraint on the robots and causes them to move along a wavy path around the desired angle. If either of the front two sensors detect an object within a threshold distance, the robot spins.

Fig. 6. (a) An example map for (b) an environment containing static and dynamic objects. The rectangular objects are static and the spheres are free to move. Map cells range in value from 0 (black) to 510 (white).

The performance of the cooperative mapping and exploration algorithms when simulated in the Webots environment are evaluated in the following section.

IV. RESULTS

This section presents the results of simulating the mapping and exploration algorithms described in Section III in Webots using the e-puck robot model described in Section II.

A. Mapping

Fig. 6a shows the resulting map for the dynamic environment shown in Fig. 6b. This demonstrates that the map is effective in dynamic environments and tends to record only static objects. It should be noted that these soccer balls generally moved fast compared to the robots. A slower moving environment would generate a map with ghosted objects that slowly faded.

B. Exploration

Simulation parameters: All simulations presented in this section were performed using the following parameters:

Map dimensions: $X \in [-0.9, 0.9]$, $Z \in [-0.9, 0.9]$, cell size: 0.01

Map range: 0 to 510

Robot charge: -10000

C (maximum cell charge): 127

The basic environment used was a $2\text{m} \times 2\text{m}$ enclosed area containing no obstacles.

1) *Efficiency of coverage*: Efficiency of coverage was measured by recording the time taken for the centre $1.8\text{m} \times 1.8\text{m}$ section of the simulation environment to be completely traversed n times by each strategy, where n ranged from 1 to the maximum number of visits. Here, the map is deemed to be

Fig. 7. Coverage of an empty $1.8\text{m} \times 1.8\text{m}$ area by 3, 5 and 10 robots for each path planning strategy. The variable n is how many times the map has been completely covered.

completely covered n times if 95% of cells have been visited n times. Fig. 7 shows the coverage results for each strategy and Fig. 8 shows the maps generated for 3 robots over time. The dynamic-step strategy consistently performed faster at all levels of map coverage. It was expected that it would perform best at low levels of coverage but slowest at exploring the environment the maximum number of times. The difference in actual performance from what was anticipated may be due to the limitations of the path execution algorithm used. As described earlier, this strategy imposes a not insignificant turning circle on the robots. Viewing the trajectories of robots in Fig. 8 shows that this impacted on the paths of the fixed-step and ramp strategies more so than the dynamic ramp strategy. This would have reduced the efficiency of these two strategies. Additionally, robots tended to disperse to the edges of the environment as it became highly explored when using the fixed-step and ramp strategies, probably due to repulsion between robots overcoming any remaining attraction from cells as average charge decreased. This may have slowed these simulations further. This may be overcome by reducing the robot charge proportionally to the average charge of map cells.

2) *Scalability*: Fig. 7 shows a comparison of the performance of the three strategies when used with 3, 5 and 10 robots. All three strategies perform significantly better with additional robots. In fact, exploration times reduced by the same factor that the number of robots increased, as can be seen in TABLE I. Means of decreases in time closely matched the increases in robots and very low standard deviations show

TABLE I
SCALABILITY OF EACH PATH PLANNING STRATEGY. INCREASES AND DECREASES GIVEN AS FACTORS.

Strategy	Increase in no. of robots	Mean decrease in exploration time	Standard deviation
DS	1.667	1.666	0.0211
DS	2	1.996	0.0157
FS	1.667	1.643	0.0213
FS	2	2.003	0.0224
R	1.667	1.695	0.0431
R	2	2.006	0.0243

Fig. 8. Maps of 3 robots using the dynamic-step path planning strategy (top), fixed-step strategy (middle) and ramp strategy (bottom). Maps are taken at 60s, 840s, and 1680s, for each column from left to right.

that this is consistent across all n . This implies that exploration times for the three strategies scale linearly with the inverse of the number of robots.

3) *Robustness to initial conditions*: Fig. 9 shows the exploration times for each strategy with the robots starting in two contrasting positions—grouped at the centre of the environment, and spread at the edges. There appears to be no significant difference in the efficiency of each strategy for each set of starting locations. While the ramp strategy does show some deviation, this is within the typical fluctuations that were observed for this strategy. This suggests that all three strategies are robust towards the initial placement of the robots within the environment.

4) *Distribution of coverage*: Fig. 10 shows the distribution of coverage for each of the three strategies. It can be seen that the distribution for the dynamic-step case peaks around one value of n for each time. This shows that all cells in the map have been visited roughly the same number of times at any given instance, indicating that the dynamic-step strategy covers the area evenly. This agrees with the appearance of the maps in Fig. 8, as well as the linear shape of the graph in Fig. 7. Contrasting this, the fixed-step strategy histograms show that this strategy does not explore evenly. At any given time, by far the greatest number of cells are either completely unexplored or completely explored, with very few in between. The maps in Fig. 8 also fit this description. This explains the slow beginning followed by a sharper increase of the curve in Fig. 7. As expected, the ramp strategy performed in between the other two strategies, showing weak peaks in Fig. 10 and semi-even coverage in Fig. 8.

5) *Robustness to objects*: In order to investigate the robustness of the exploration algorithm to objects, it was simulated with a single obstacle introduced to the environment. As

Fig. 9. Coverage of an empty $1.8\text{m} \times 1.8\text{m}$ area by 3 robots for each path planning strategy. Robots were initially placed either in the centre of the environment or spread around the edges. The variable n is how many times the map has been completely covered.

Fig. 10. Distribution of coverage of 3 robots using the dynamic-step strategy (top), fixed-step strategy (middle) and ramp strategy (bottom). Distributions are taken at 60s, 840s, and 1680s, for each column from left to right.

anticipated, after some time, for all three strategies, robots tended to orbit the object perpetually and ceased exploring the environment. The area inside the object remained unexplored and hence fully charged so attracted robots towards it. This indicates that a more sophisticated form of object identification that recognises closed edges and identifies some areas as non-explorable is necessary.

V. CONCLUSIONS AND FUTURE WORK

A probabilistic, cell-based, multi-agent map making and merging scheme has been proposed and has been shown in simulation to be effective in representing dynamic environments. A potential field based exploration algorithm has also been presented with three variations—dynamic-step, fixed-step, and ramp. It was demonstrated that the dynamic-step strategy is most effective at covering the environment

both quickly and evenly. Additionally, the efficiency of all three strategies scale linearly with the number of robots and are robust to the initial placement of robots. The presented exploration algorithm is designed for exploring open spaces and consequently does not effectively deal with objects in the environment. An extension to this algorithm could use the object information in the map to identify continuous edges and closed objects, and then declare the area inside the object neutrally charged. The performance of the exploration algorithm could also be improved by a more advanced path execution algorithm. Future work will concentrate on implementing these algorithms in a testbed as a remote controlled co-simulation with virtual and real robots.

ACKNOWLEDGEMENTS

C. Yu is the recipient of an Australian Research Council Postdoctoral Fellowship (project number DP-0877562).

REFERENCES

- [1] M. Polycarpou, Y. Yang, and K. Passino, "A cooperative search framework for distributed agents," *Intelligent Control, 2001. Proceedings of the 2001 IEEE International Symposium on*, pp. 1–6, 2001.
- [2] D. Pack and B. Mullins, "Toward finding an universal search algorithm for swarm robots," *Intelligent Robots and Systems, 2003. Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, pp. 1945–1950, Oct. 2003.
- [3] J. Riehl, G. Collins, and J. Hespanha, "Cooperative graph-based model predictive search," *Decision and Control, 2007 46th IEEE Conference on*, pp. 2998–3004, Dec. 2007.
- [4] L. Giannetti and P. Valigi, "Collaboration among members of a team: a heuristic strategy for multi-robot exploration," *Control and Automation, 2006. 14th Mediterranean Conference on*, pp. 1–6, June 2006.
- [5] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," *Robotics and Automation, 2000. Proceedings. IEEE International Conference on*, vol. 1, pp. 476–481, 2000.
- [6] D. O'Beirne and M. Schukat, "Collaborative exploration for a group of self-interested robots," *Computer and Robot Vision, 2005. Proceedings. The 2nd Canadian Conference on*, pp. 184–191, May 2005.
- [7] B. Yamauchi, "A frontier-based approach for autonomous exploration," *Computational Intelligence in Robotics and Automation, 1997. Proceedings. IEEE International Symposium on*, pp. 146–151, Jul 1997.
- [8] R. Zlot, A. Stentz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," *Robotics and Automation, 2002. Proceedings. IEEE International Conference on*, vol. 3, pp. 3016–3023, 2002.
- [9] J. Stipes, R. Hawthorne, D. Scheidt, and D. Pacifico, "Cooperative localization and mapping," *Networking, Sensing and Control, 2006. Proceedings. IEEE International Conference on*, pp. 596–601, April 2006.
- [10] J. Vazquez and C. Malcolm, "Distributed multirobot exploration maintaining a mobile network," *Intelligent Systems, 2004. Proceedings. 2nd International IEEE Conference on*, vol. 3, pp. 113–118, June 2004.
- [11] E. Saad, M. Awadalla, A. Hamdy, and H. Ali, "Robot formations using local sensing and limited range vision," *National Radio Science Conference, 2008.*, pp. 1–8, March 2008.
- [12] S. Sarel, T. Balch, and N. Erdogan, "Incremental multi-robot task selection for resource constrained and interrelated tasks," *Intelligent Robots and Systems, 2007. IEEE/RSJ International Conference on*, pp. 2314–2319, October 2007.
- [13] (2008) Cyberbotics Ltd. website. [Online]. Available: <http://www.cyberbotics.com/>
- [14] (2008) Using the Matlab engine to call Matlab software from C and Fortran programs. [Online]. Available: http://www.mathworks.com/access/helpdesk/help/techdoc/matlab_external/f29148.html
- [15] (2008) Matlab C and Fortran API reference. [Online]. Available: <http://www.mathworks.com/access/helpdesk/help/techdoc/apiref/bqoqnz0.html>