Local Navigation Strategies for Multi-Robot Exploration: From Simulation to Experimentation with Mini-Robots

Andry Tanotoa, Ulrich Rückertb

aSystem and Circuit Technology, Heinz Nixdorf Institute, University of Paderborn, Fürstenallee 11, Paderborn 33102
bCognitronics and Sensor Systems Group, Cognitive Interaction Technology Centre of Excellence (CITEC) Bielefeld University, 33615 Bielefeld, Germany

Abstract

This paper presents two local navigation strategies for multi-robot systems in performing exploration of unknown environments. The strategies are based on the frontier cell for mapping the environment and navigating in it. Additionally, two coordination strategies are used to solve possible conflicts among robots. Key criteria investigated are the efficiency and effectiveness represented by the completion time, travelled distance, the total steps, and the task distribution. The developed algorithms are tested in simulations as well as in experiments with Khepera III mini robots running on the Teleworkbench. To evaluate the robustness of the developed algorithms, tests are performed under different environmental configurations and varying numbers of robots. The results show the advantages of both proposed strategies in different situations such as environment types and starting positions.

Keywords: multi-robot system; environment exploration; mini robot; teleworkbench; robot experiment

1. Introduction

Autonomous exploration of unknown environments is one of the fundamental aspects in robotics for successful robot deployment in real applications such as search operation in dangerous areas, e.g. burning buildings or mine fields [1, 2]. In exploration tasks, map building is necessary for the robots to update their knowledge of the environment and simultaneously navigate in it. In such an application, multi-robot systems can be advantageous as exploration tasks are parallelisable. Additionally, multi-robot systems are more robust to failures of one or more robots. In multi-robot systems, task assignment is an important aspect for avoiding redundancies and conflicts [3] or for increasing efficiency [4]. Another important aspect is the communication among robots. The more efficient the communication is, the more scalable the coordination is [5].

This paper presents two strategies for a multi-robot system to explore its environment: an extension to the modified local navigation algorithm [6] and go-straight with segmentation. The strategies are based on local navigation with
which the robots move to their cheapest frontier-cells, which are defined as cells that are located at the boundary between explored and unexplored areas. Additionally, the strategies are intended for short-sighted robots that can only sense nearby surrounding objects. Hence, the robots must visit every corner of the environment.

We will first test the developed algorithms in a robot simulator. Afterwards, we conduct some experiments using mini robots to further verify the algorithms. Experimentation is done using the Teleworkbench [7], which can support us in setting up, executing, observing, and analysing the experiments. Additionally, the environment models used in the simulator can be ported into real environments with the help of the automatic environment building module of the Teleworkbench. Thus, we can compare the results from the simulation and the experimentation with real robots.

The paper is organized as follows. In Section 2, we explain the proposed exploration algorithms. Section 3 presents the implementation of the proposed algorithms along with the description of the software and hardware used. The deployment scenario and test results are presented in Section 4. In the end, Section 5 concludes this paper.

2. Multi-Robot Exploration

2.1. Grid-Based Environment Exploration Algorithm

In grid-based mapping, the environment is divided into uniform sized areas called cells. Each cell can take different states, depending on the value that the robot’s sensors deliver. There are two types of grid maps (see Fig. 1a) that differ on how the cell states are represented: coverage or occupancy grid maps. In the occupancy grid map, each cell state is represented in a percentage of the object coverage in the cell. However, the cells in the occupancy grid map are considered as either free or occupied.

2.1.1. Modified Local Navigation Algorithm (MLNA)

Amin et. al. present the MLNA and its implementation on the Khepera II mini robot [6]. The MLNA uses an occupancy grid map and assigns one of four possible states (visited, object, unexplored, and frontier) to each cell. The algorithm computes the next step the robot has to take based on the cost of reaching the free adjacent cells. The cost function $C$ for a free cell $P$ is given as:

$$C(P) = N(P)$$

(1)

Where $N(P)$ is a function for computing the number of free cells adjacent to cell $P$.

The modification of the algorithm for supporting exploration using more than one robot is presented in [8]. This paper presents an extension to the aforementioned modified algorithm through the application of coordination vectors which will be further detailed in Subsection 2.2.
2.1.2. Go-Straight Algorithm

This algorithm controls the robot to move straight ahead and turn when it encounters an object. The turn direction is determined based on the existence of objects on either side of the robot detected by the range sensor. If no object is detected on both sides, the robot turns to the left.

2.2. Multi-Robot Coordination

To avoid conflicts during runtime, coordination among robots is needed. The following paragraphs explain two coordination strategies applied on one of the previous two exploration algorithms.

Coordination Vector. To minimize conflicts among robots during runtime, we introduce a vector that a robot should follow if it has to choose between two or more possible actions. We call this vector a coordination vector that will force the robots to separate from each other, thus conflicts can be minimized. The coordination vector is calculated by adding all vectors, whose size and orientation depend on the position of the current robot and the others. In our case, the vector size is proportional to the distance between two robots. Thus, a nearby robot generates higher repelling forces than a farther robot. Fig. 2a depicts the calculated vectors applied on the robots and how they are used in selecting the next cells to traverse.

Segmentation. Another possible approach of coordination is by partitioning areas where the robots operate [9]. Each robot will be assigned a segment in the environment to explore and there are two or more overlapping segments. The segment is generated dynamically during runtime; if one robot completes a segment, it will go to a free area and build another segment. Fig. 2b shows an example of the segmentation approach in the case of two robots. Due to communication delay, a conflict can occur in which two segments are overlapping. Two solutions are proposed to solve this problem. In the case of a different segment size, the smaller segment gets the intersecting area. Should the segments have the same size, the conflict is resolved by using the identification number (ID) that each robot gets during initialization. The robot with smaller ID claims the overlapping area.

3. Implementation

This section describes the implementation of the presented algorithms. But first, we will see some supporting hardware and software that are used in the implementation.

Teleworkbench. The Teleworkbench [7, 10] is a teleoperated platform for conducting experiments with single- or multi-robot systems. It can provide a controlled environment in which different algorithms can be tested and com-
pared. Some features of the Teleworkbench are the support of up-to four parallel experiments, wireless robot communication (WLAN and Bluetooth), automatic environment building, a robot positioning system for up to 64 robots, robot message logging, a web-based user interface, and live-video streams of the experiments. Additionally, an 18 TB file server is provided to store all the generated data.

**Khepera III.** Khepera III [11] is the successor of the Khepera II minirobot. In comparison with its predecessor, the Khepera III is more than twice as big. The base module is equipped with infra-red and ultrasonic sensors. Additionally, it can be extended by a Korebot II board which has a 600 MHz processor and an embedded Linux operating system.

**Player/Stage.** As the robot programming framework, we use Player/Stage [4]. Player provides an abstraction of robot hardware that offers us a unified access to different hardware components of the same type. The robot simulator Stage is one important component of the framework. We can simulate the robots interacting with the environment using different sensors. One advantage of this framework is that the Player Client can be used to connect to either the actual robot or the one on the simulator.

**Environment Configuration.** Two environment configurations are used for testing the developed algorithm: room and blocks (see Fig. 3). The room-like environment has more areas with multi-rows and multi-columns while the block-like environment consists predominantly of areas with either single-rows or single-columns.

**Robot Communication.** In our implementation, we use a blackboard as a means for all the robots to exchange information with each other. There is some information shared among robots such as the current robot position, global map, current cell along with its neighbouring cells, and the target cell (see Fig. 4). The robot position information of other robots is used for preventing the robot to mark a cell as occupied when another robot is detected by the range sensor.

<table>
<thead>
<tr>
<th>Header</th>
<th>Robot ID</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q: Request for a master grid maps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P: Actual robot position</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U: Update Cell(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S: Segment data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4: The structure of the data packet for inter-robot communication. Four types of messages indicated by the header are supported.
Fig. 5: A diagram depicting system deployment on the Stage simulator (left) as well as the Teleworkbench (right).

**Metrics.** There are some aspects that we would like to measure from the developed algorithms: completion time, task distribution, total steps, and total travelled distance. The completion time is the time needed by the robots to complete the exploration which is indicated when there is no free and accessible cell in the environment. The task distribution represents the performance of the strategy in distributing the task to each robot. This value is derived by calculating the distance between a perfect task distribution (1.0) and the standard deviation of the percentage of tasks completed by the robots. The total steps is calculated by summing all steps taken by the robots during exploration. The total travelled distance is derived from summing the distance the wheels of each robot travel during runtime. We take the wheel travelled distance into consideration as we deem that the total rotation of the robots can contribute to the total distance.

4. Test and Result

4.1. Deployment

The first test phase uses the Stage robot simulator that is deployed on a host PC with the robot controllers running on another PC (see Fig. 5 left). The robot controllers communicate with the Stage simulator over the network using TCP/IP and each controller is responsible for controlling one robot. Three different starting points are used on each environment configuration.

The second test phase involves the Teleworkbench, which provides us with precise robot position and orientation as well as event and message logging. The robot programs are first compiled using a cross-compiler before being deployed on the robots. Robots are connected to one server (hereafter called the gateway) of the Teleworkbench over a WLAN router which provides a private network (see Fig. 5 right). Through this, the robots can acquire their position from the Teleworkbench Server.

4.2. Results

Fig. 6 shows the test results using the Stage simulator and the Khepera III mini robots on the Teleworkbench. There are no results involving one robot with the Teleworkbench as we will compare the results of simulation and experiment only for multi-robot exploration. From the simulation results, we can see that the straight with segmentation algorithm is faster than MLNA with coordination vector (hereafter called EMLNA). However, the results of experiments with real robots show otherwise (Blocks 2, Blocks 3, and Room 3). These cases are caused by the failure of the real robot in detecting the wall at the corner, which did not occur during the simulations. As a result, the robots have to travel back to the cells with an undetected wall.

In general, the use of more robots decreases the exploration time, although the speed-up is not necessarily linear. In the simulation with two robots, the achieved speed-up is in the range from 1.56 (Room 2 with EMLNA) to 2.09 (Blocks 2 with EMLNA). In the case of simulation with three robots, the speed-up is from 2.38 (Room 3 with EMLNA) to 2.74 (Blocks 3 Straight). Results from the experiments in the Teleworkbench show that the increasing number of robots from two to three almost gives a linear speed-up in all cases except for the case of the room type environment.
with the Straight algorithm (only 1.1 speed-up).

In terms of task distribution, the results from the simulation and the Teleworkbench are consistent in most cases except for the case of exploration in blocks environment with two robots (Blocks 2). The simulation results show that EMLNA has a better performance in comparison with the Straight algorithm. However, the results of the experiments show otherwise.

The total steps and total distance are in general consistent. It is only in the case of the blocks environment with one robot and the room environment with three robots that we can see that an increase in the total steps value does not lead to a higher total distance and vice versa. This can be determined by the paths of robots, whether they are mostly straight-forward or comprise many turns and diagonal paths. Fig. 7 shows the visualization of the exploration during runtime using our graphical user interface. The dark areas represent unexplored cells while grey areas are adjacent unvisited cells. Open areas are visited cells, thus we can see in there the visualized robot paths.

5. Conclusion

We have presented two local navigation algorithms implemented on a multi-robot system performing an unknown environment exploration. The algorithms are tested on the Stage robot simulator and also on Khepera III mini robots running on the Teleworkbench. The test results show that the two proposed algorithms perform comparably the same. In most cases, straight with segmentation performs better in terms of completion time, however with the trade-off of a longer travelled distance or lower efficiency in task distribution.

In the near future, we will test the proposed algorithms with higher number of robots in larger environments.
Fig. 7: The snapshots of the recorded video overlaid by the grid map and the robot path at the beginning (left) and at the end (right). The walls are built using white plastic blocks that are placed automatically by a gripper module.

Additionally, a comparison of other similar algorithms will be done to demonstrate the advantage of the Teleworkbench as a test-bed for multi-robot systems.

Acknowledgements

This work has been supported within the Excellence Initiative of the German Research Foundation (DFG) by the Center of Excellence “Cognitive Interaction Technology” (EXC277), by the DFG Collaborative Research Center ”Self-Optimizing concepts and structures in mechanical engineering” (SFB614), and by the DFG grant (INST 214/47-1 FUGG).

References