ExmoR – A Testing Tool for Control Algorithms on Mobile Robots

F. Lehmann, M. Ritzschke and B. Meffert

Institute of Informatics,

Humboldt University, Unter den Linden 6, 10099 Berlin, Germany

E-mail: falk.lehmann@gmx.de, ritzschk@informatik.hu-berlin.de

Abstract. This paper describes the testing tool ”ExmoR” that allows the user to experiment with different control algorithms as map building and a routine to explore the environment. The intention is to give the user the possibility of easy testing the various control algorithms for the Khepera robot without any knowledge about the robot language, the control architecture or other robot details.

1 Introduction

The aim of the project ExmoR [1] – experiments with mobile robots – is to provide students or beginners in robotics with an easy tool for testing control algorithms on a real robot. Known existing tools are often either only simulators or assume knowledge about programming languages (for example [2]).
The basic version of a Khepera mini-robot is used to perform the movements [3]. This robot has only 8 infrared proximity sensors with limited range.

2 Control architecture

The heart of the program is a behaviour oriented control architecture as described in [4]. It allows the user to modify a certain level without revising the complete control structure. The control architecture consists basically of 4 levels as shown in Figure 1.

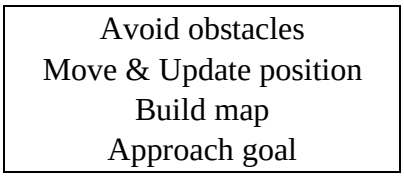

Figure 1. The 4 basic behaviour levels in ExmoR

The first level of the architecture is used by the robot to avoid obstacles. Sensor inputs are transformed into speed instructions by a decision tree or by fuzzy control. In the second level the robot moves at constant speed straight ahead. At the same time the relative position of the robot is calculated by odometry. The position is required by the third level where an environment map is built. The last basic behaviour manoeuvres the robot to a given goal. The goal is defined by relative coordinates and the robot’s line of vision at the goal.
Every upper level uses the data and functionality of the lower levels. For instance a map building is always coupled with obstacle avoidance and position calculation.

The basic levels are extended by a fifth level. This level manages the exploration.

3 Experimental possibilities

3.1 General options

The user can choose among various options for each level of behaviour, for instance change the distance meanings of the sensors and the goal (Figure 2.) and the velocity meanings of the motors.

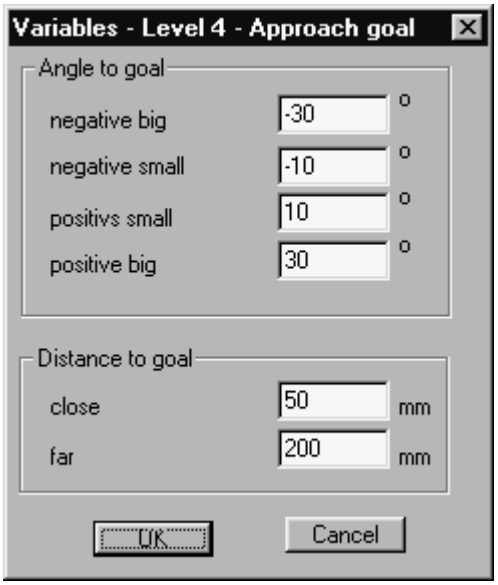

Figure 2. Example for an option dialogue

To simplify the decision trees and the fuzzy controllers all possible input and output data are limited to certain values. These values are named for better handling. For instance the distance ”close” means a range of 50 mm (Figure 2.). All these values are changeable within the program.
For the movement levels it is also possible to choose between a decision tree or a fuzzy controller.

3.2 Map building

The program supports different kinds of map building algorithms and map storing techniques. This enables the user to follow the path of the robot via screen (Figure 3).

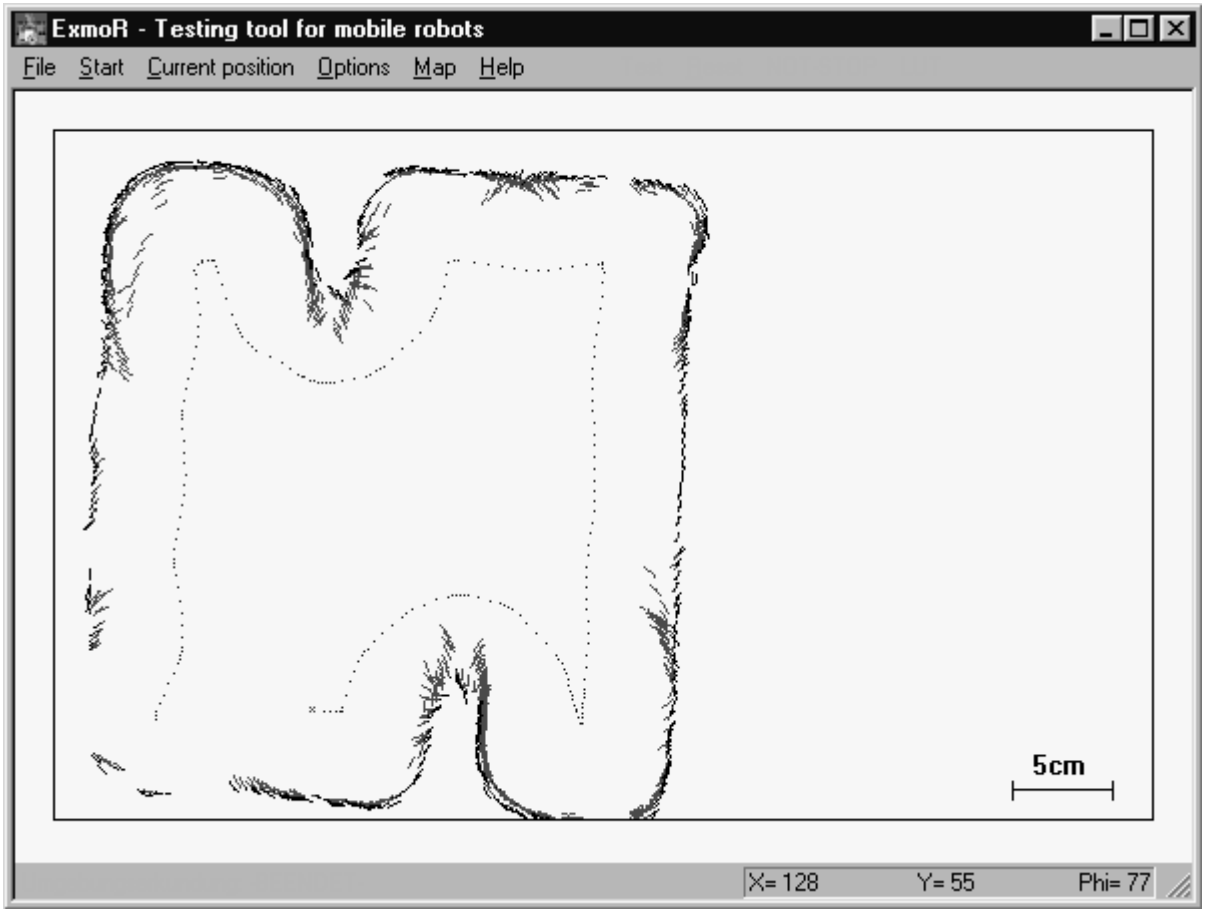

Figure 3. Visualisation of the robot's path and recognised obstacles (the robot starts at the left bottom corner)

Three different map building techniques were implemented [5] [6] [7]. Each method deals with uncertainties in sensor data and uses grid based maps. A map has either a static size or is dynamically stored. The user has also to choose the sensors which are used for map building. To increase the accuracy of the maps the characteristic values of the sensors were measured under various conditions. The results of these measurements show that light, heat, the surface of the obstacle, the surface of the ground and nearby fluorescent lamps influence the characteristics of the sensors. An example for two different brightness levels is given in Table 1. The conclusion of the experiments is that it is very important to keep these environment conditions constant to get exact maps.

Location of the infrared sensors	Left 90°	Left 45°	Left 10°	Right 10°	Right 45°	Right 90°	Right 170°	Left 170°
10 bit sensor value at darkness	232	579	437	250	2	748	692	705
10 bit sensor value at daylight	352	689	534	352	13	848	795	799

Table 1. Average sensor values as received from the A/D converter at different brightness levels (obstacle distance 4 cm)

Another problem of the map building is the accuracy of the position. The possibilities of the Khepera robot are very limited compared to an extern determination of the position.

3.3 Complete exploration of the environment

For the exploration of an unknown closed environment a special algorithm was implemented which allows the robot to find all outer walls and inner obstacles in a short time (Figure 4.). The environment is divided into virtual sectors. These sectors comprise a rectangle area. For every pair of coordinates the sector can be calculated. The larger the sector, the more pairs of coordinates are included. The algorithm requires that the robot visits every reachable sector of the working space. The size of the sectors is an important factor for the exploration. The smaller the sectors are, the more sectors the robot has to visit and the more detailed the map will be.

1. Start at outer wall
2. Follow outer wall till start position is reached, save all visited sectors
3. Calculate maximal extension of the travelled space
4. Create array with this extension
5. Mark all sectors from step 2 as ”obstacle” in this array
6. Mark all unmarked outer sectors and appending sectors as ”unreachable”
7. Navigate to the next unmarked sector and mark sector as ”free”; if an unmarked obstacle is detected continue with step 8, else continue with step 7 until all sectors are marked; in that case stop
8. Move completely around the obstacle and save the visited sectors
9. Calculate maximal extension of the obstacle
10. Mark all sectors from step 8 as ”obstacle” and ”temporary”
11. Mark all outer sectors from step 9 and appending sectors as ”temporary”
12. Mark all sectors from step 9 without ”temporary” marking as ”unreachable”
13. Remove ”temporary” marking
14. continue with step 7

Figure 4. Course of events of the exploration algorithm

The algorithm is theoretically capable of exploring all kinds of environments and artificial labyrinths. The only problem is, as in map building, the position accuracy. Therefore the robot sometimes surrounds already visited obstacles or the outer wall. An example of a complete exploration is shown in Figure 5.

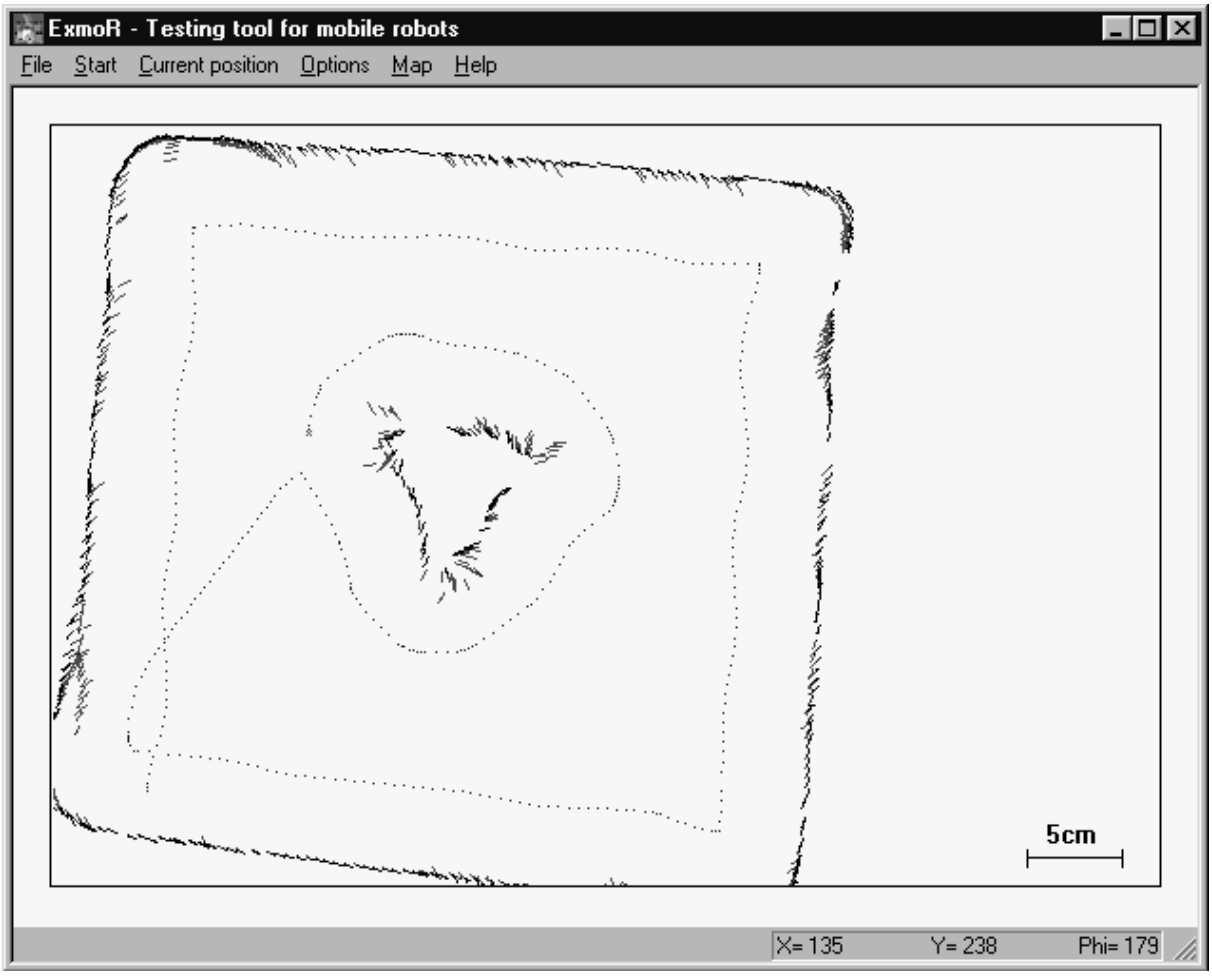

Figure 5. Example of an explored environment with 1 obstacle in the middle (the robot starts at the left bottom corner)

3.4 Control algorithm

In each movement level of the robot architecture it is possible to operate with a decision tree or with a fuzzy controller.

❑ Decision tree

One of the main goals of the project was to create a tool for algorithm testing without any knowledge about robot details and the program source code. Therefore a decision tree interpreter was implemented that allows the user to modify and test out control algorithms at runtime. The necessary rules were formatted in an easy artificial script language (Figure 6.)

```
0:  ir0<far, ir1<far;   v=0, phi=l_slow;   #1
1:  ir4<far, ir5<far;   v=0, phi=r_slow;   v=fast, phi=0
```

Figure 6. Example of a rules set for a decision tree

One rule consists of 4 parts. Part 1 is the serial number used for jump operations. Part 2 contains the condition. If the condition is true the 3. part is executed, else the last part. The 3. and 4. part can either contain velocity information or a jump statement.

There are two ways to provide the program with a rules set. One way is the direct input of rules in the program, the other is creating ASCII files. This makes the user more independent of the program. Students can develop decision trees at home and test them with the real robot and the tool.
After implementing the rules set it is transformed into an internal format. This format allows easy access and evaluation of the rules and conditions. The re-transformation of the internal format into human readable rules is also possible.

❑ Fuzzy controller
The source code for the fuzzy controller is generated by a fuzzy tool. Any changes of the controller require complete recompiling. To eliminate this disadvantage it is planned to use the fuzzy tool as an extern fuzzy controller which is running simultaneously with the ExmoR program.

4 Example

First of all the tool was developed for students to learn by experiments more about control algorithms for mobile robots. The following could be an exercise for students at home.
Create a decision tree for level 4 (approach goal) with the following conditions:

1. reach any goal in shortest possible time
2. use minimal number of rules

The student's solutions could be compared and discussed. Figures 7. and 8. show two examples for possible solutions.

```
0:  alpha>pos_small;   v=0, phi=r_fast;   #1
1:  alpha<neg_small;   v=0, phi=l_fast;   v=fast, phi=0
```

Figure 7a. Example 1: Turn against the goal and move straight ahead

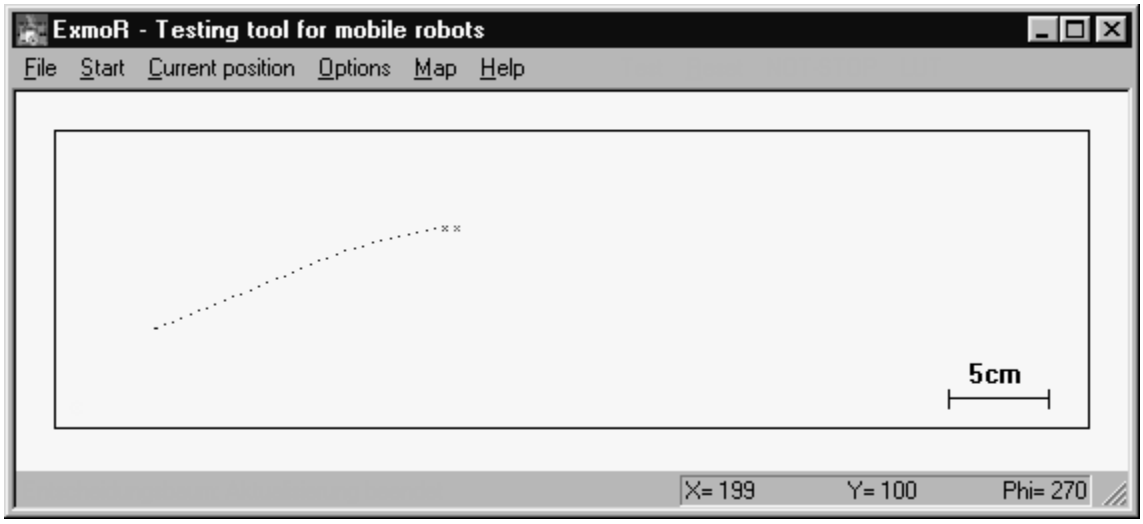

Figure 7b. Robot's path for example 1

```
0:  rho<near;   #1;   #3
1:  alpha>pos_small;   v=0, phi=r_fast;   #2
2:  alpha<neg_small;   v=0, phi=l_fast;   v=fast, phi=0
3:  alpha>pos_small;   v=slow, phi=r_fast;   #4
4:  alpha<neg_small;   v=slow, phi=l_fast;   v=fast, phi=0
```

Figure 8a. Example 2: Already move while turning against the goal

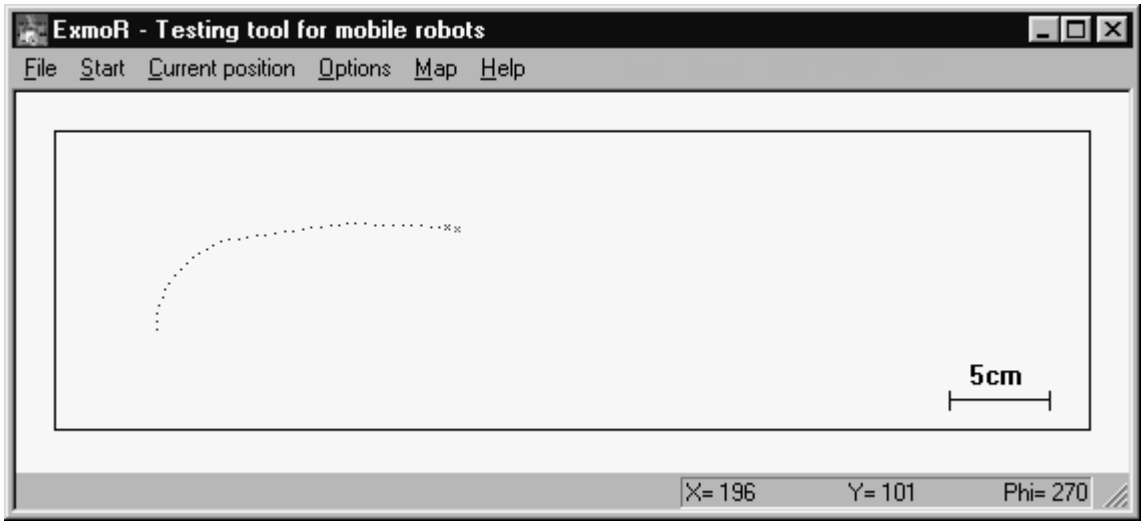

Figure 8b. Robot's path for example 2

In a test run with example 1 the robot needs about 8 s to reach a specified goal, with example 2 about 6.5 s for the same goal.

Summary

The applied tool is realized in C++ and running under windows 95/98. It has a user friendly interface to carry out different tests. It is possible to implement a rule set for the control of the robot and to test the quality of the algorithms on-line. Another feature of the object oriented realisation is the comfortable possibility to make extensions. It allows the user to implement his own algorithms on all behaviour levels.

References

[1] **F. Lehmann**: ExmoR - Ein Experimentierplatz für mobile Roboter, *Diplomarbeit*, Humboldt-Universität Berlin, Institut für Informatik, 1999

[2] **O. Michel**: Khepera Simulator version 2.0, *User Manual*, University of Nice 1996

[3] **F. Mondada, E. Franzi and P. Ienne**: Mobile robot miniaturisation: A tool for investigation in control algorithms. *Experimental Robotics III, Proceedings of the 3rd International Symposium on Experimental Robotics*, Springer Verlag 1994, London 1994, pp 501-513.

[4] **R.A. Brooks**: A robust layered control system for a mobile robot, *IEEE Transactions on Robotics and Automation*, Vol. RA-**2**, No.**1**, March 1986, pages 14-23.

[5] **A. Elfes**: Sonar-based real-world mapping and navigation, *IEEE Transactions on Robotics and Automation*, Vol.**3**, No.**3**, 1987, pages 249-265

[6] **M. Beckerman and E.M. Oblow**: Treatment of systematic errors in the processing of wide-angle sonar sensor data for robotic navigation, *IEEE Transactions on Robotics and Automation*, Vol.**6**, No.**2**, April 1990

[7] **J. Borenstein and Y. Koren**: Real-time map-building for fast mobile robot obstacle avoidance, *SPIE Vol.**1388**, Mobile Robots* **V**, 1990