In Proceedings of the 5th International European Conference on Artificial Life (ECAL99), volume 1674 of Lecture Notes in Artificial Intelligence, pages 329-333, Lausanne, Switzerland, 13. - 17. September 1999. Springer-Verlag.

A Visualization Tool for the Mini-Robot Khepera: Behavior Analysis and Optimization

Axel Löffler, Jürgen Klahold, Manfred Hußmann, and Ulrich Rückert

Heinz Nixdorf Institute, System and Circuit Technology, Paderborn University,
Fürstenalle 11, D-33102 Paderborn, Germany,
loeffler@hni.uni-paderborn.de,
http://wwwhni.uni-paderborn.de/fachgruppen/rueckert/KHEPERA

Abstract. The design of behavior generating control structures for real robots acting autonomously in a real and changing environment is a complex task. This is in particular true with respect to the debugging process, the documentation of the encountered behavior, its quantitative analysis and the final evaluation. To successfully implement such a behavior, it is vital to couple the synthesis on a simulator and the experiment on a real robot with a thorough analysis. The available simulator tools in general only allow behavioral snapshots and do not provide the option of online interference. In order to cure these shortcomings, a visualization tool for *aposteriori* graphical analysis of recorded data sets which gives access to all relevant internal states and parameters of the system is presented. The mini-robot Khepera has been chosen as experimentatory platform.

1 Introduction

The design of behavior generating control structures for real robots acting autonomously in a real and changing environment is a complex task. In [12] the usefulness of embodiment for robotics is comprehensively pointed out as algorithms developed by sole simulation of autonomous agents in restricted and controlled environments may fail when transfered to a real system. Nevertheless, simulators as in [9] are very useful to obtain a primal executable version of a control structure generating the desired behavior. On the other hand, especially due to the fact that, in general, only momentary snapshots of the encountered behavior are available through these tools, a judgment of the behavioral dynamics is made very difficult. Moreover, the option of online interference concerning for example parameter variations is naturally minimal. For this, a completely new test run is required without the possibility to compare two instances directly. Coupling of synthesis and analysis in a feedback loop leads to an evolutionary process between implemented behavior and designer's knowledge. Therefore, we propose to extend the common two step program in behavior design (create a satisfying simulator solution, then adapt it to the real robot case) by a third stage, an *aposteriori* holistic analysis of the encountered behavior (Fig. 1). The latter requires the development of appropriate software tools as the one presented in this paper. Our basic approach for the implementation of behavior generating control structures for autonomous agents follows the broad outline of the so-called *nouvelle Artificial Intelligence* [2]. Complex behavior is produced by the interaction

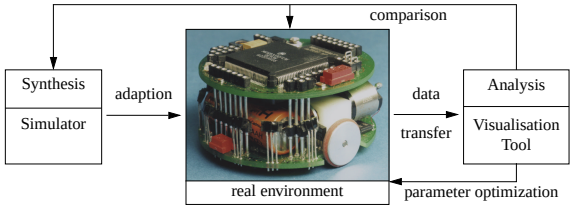

Fig. 1. The proposed design cycle with a photography of the mini-robot Khepera [3].

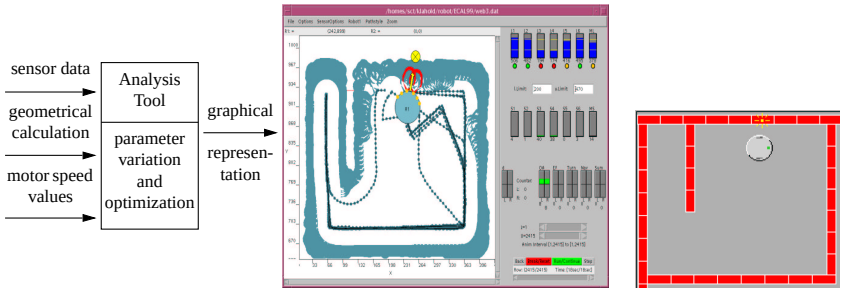

Fig. 2. A schematic of the input data flow of the visualization tool with a full-scale view of its graphical representation is shown together with the corresponding simulated environment ($40\,cm \times 35\,cm$, total runtime: 18 sec). The control panel (bottom right) of the visualization tool permits to select a specific part of the data set in question, to start or stop the visualization at a particular point and to view the data stepwise.

of 'simple' modules active in parallel like the Braitenberg patterns [1], artificial neural networks, geometric calculations and rule-based reasoning without relying on extensive and resource inefficient internal representations of the environment. The basic modules are either hardwired or adaptive. In the first case, one may rely on genetic algorithms and evolutionary programming [4]. An alternative is to consider hand-designed algorithms, wherein the optimization process is left to the designer. This approach usually leads to the design of a more complex and easier-to-understand behavior. Secondly, one may endeavor the implementation of adaptive, self-organizing structures. Here, either online learning [7], [13], which allows to adapt to an unknown environment, or off-line training [8], which - although limited in the case of a changing environment - in general produces more accurate results, are conceivable. Eventually, [5], [6] and [10] may be seen as complementary to the presented approach. Firstly, evolutionary robotics instead of hand design are used to build the controllers. Secondly, the reality gap between simulation and real robot behavior is bridged not by *aposteriori* analysis, but by an *apriori* set up of valid simulations. Common to all these approaches is the need for data documentation permitting a quantitative analysis, which has prior only been possible through simulations [11]. By means of the graphically represented data, an evaluation of the encountered behavior becomes possible. Hence, the designer is able to conduct an optimization process by appropriately varying the system and training parameters without directing a considerable number of test runs. The visualization tool (Fig. 2) described in the following section is conceived to actively support this process.

2 Building a dynamical view of the environment

The main purpose of the presented software tool is to build a holistic view of the robot's environment, which may be subject to dynamical changes. Note that the perceived distribution of obstacles is purely subjective, i.e. derived from the sensory-motor data of the robot. To stay abreast of environmental changes, special features (Fig. 2) permit to suppress obsolete parts of the encountered obstacle history. Every navigational tasks like transport or homing is based on a position calculating process. Since the mini-robot Khepera in its basic configuration does not dispose of far-ranging active sensors necessary to obtain topological information, we apply an explicitly geometrical method. The current position is obtained by odometrical path integration using the incremental encoder values n_L, n_R as variables and the wheel distance d as well as the advancement per pulse Δl as system parameters. Since the designer knows the exact geometry of the real environment, one may compare the robot's perception to reality. This is demonstrated by an experiment using an environment of rectangular geometry ($75\,cm \times 60\,cm$). The robot follows the walls for roughly two rounds (92 sec, i.e. 1126 program cycles) and stops at an internal angle of 71.7° instead of 90° (correct). The following results concerning angular errors due to parameter variations (original ones: d=52mm, Δl=0.08mm) have been obtained by means of the visualization tool (Table 1,2):

Δl [mm]	0.076	0.077	0.078	0.079	0.080	0.081	0.082	0.083	0.084
$\Delta\alpha$ [°]	18.6	9.4	0.1	-9.1	-18.3	-27.6	-36.8	-46.0	-55.2

Table 1. Angular error$\Delta\alpha$ by variation of Δl.

d [mm]	48	49	50	51	52	53	54	55	56
$\Delta\alpha$ [°]	-79.8	-63.5	-47.9	-32.8	-18.3	-4.4	9.0	21.9	34.4

Table 2. Angular error $\Delta\alpha$ by variation of d.

Thus the presented tool may be used to optimize the odometrical parameters for individual real robots. Moreover, it gives the designer a hint of the time interval during which the position calculation system works sufficiently well. This may be seen through another experiment, where the same environment as before, but with an additional light source has been used. An *event* is defined by locating a light source. After the first detection, the robot was able to recognize the light source five times, then registered it as a different one (a confidence area of 20cm in Manhatten distance concerning the light source position was used). Hence, the positioning system worked well during 297 sec, i.e. 3825 update cycles. As already mentioned, the infrared proximity sensors are used to construct an estimation of the current environmental structures. Moreover, these data also form the sensor space on which different exploration modules like obstacle avoidance (OA), edge following (EF), turning (Turn) and point-to-point navigation (Nav) work in parallel, which are visualized in the control panel (Fig. 3). The spatial form of the sensor characteristics may be customized in order to simulate sensor degradation or the use of other kinds of proximity sensors. Furthermore, an artificial neural network has been applied for extracting a symbolic angle-to-light source information from the subsymbolic sensor data stream (Fig. 4). Eventually, a comparison between the simulator (Fig. 2) solution and

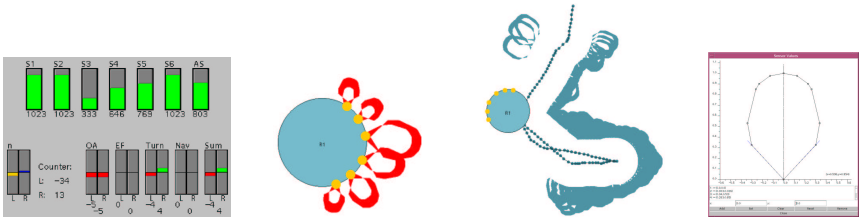

Fig. 3. Visualization of the infrared sensor values (S1-6, A[verage]S), the speed values determined by different software modules and the values from the incremental encoders n (left). The possible positions of obstacles, which are causing this specific sensor input, are drawn around the robot (middle). A high sensor value corresponds to a nearby and unextended obstacle whereas a low one indicates a dislodged object. The average sensor noise is taken into account by the thickness of the traits. Finally, the sensor trait may be customized (right).

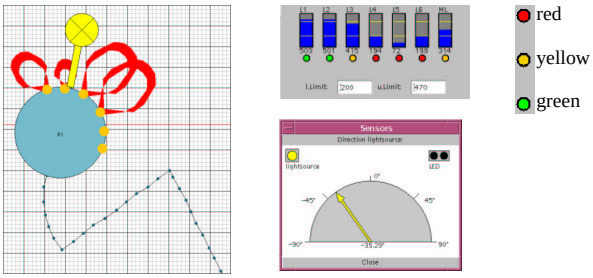

Fig. 4. Visualization of the light sensor values (top middle) and the artificial neural network (bottom middle). It is possible to feed different thresholds (top middle/right) for analyzing the photo-sensitivity of the implemented algorithms.

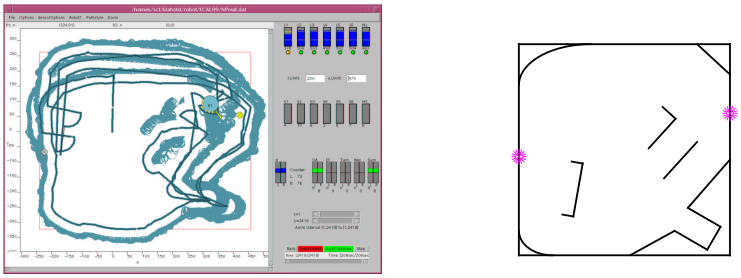

Fig. 5. A complete data set obtained from a real Khepera acting in a real environment is shown with a schematic of the corresponding actual obstacle structure ($80\,cm \times 80\,cm$, total runtime: 208 sec).

the real robot's implementation (Fig. 5) of the Dynamical Nightwatch's Problem [8] reveals a shortcoming that is inherent to all simulations: the position determination is supposed to be perfect, which is necessary for correct representation. Note that in simulations, path integration is usually not used. In contrast, for real systems, it still presents the main position calculating method. Hence, the most restricting limitation to the real robot's performance arises from the erroneous position calculation due to wheel slippage and unknown fabrication tolerances. As demonstrated above, the presented visualization tool is particularly apt to assist in the minimization of errors due to the latter.

3 Conclusion

The presented visualization tool for the mini-robot Khepera permits the extension of the common two step design program for behavior generation by a third stage: an *aposteriori* analysis of the encountered behavior. This allows to set up a synthesis-analysis feedback loop evolutionarily improving the envisaged behavioral design. In particular, the holistic visualization of the encountered behavior enables the designer to thoroughly document, analyze, evaluate and compare the performance of his implementation.

Acknowledgments. We are very grateful to Prof. G. Domik whose lecture on visualization motivated us to undertake the presented work. In addition, the corresponding author is supported by the DFG-Graduiertenkolleg "Parallele Rechnernetzwerke in der Produktionstechnik", GRK 124/2-96.

References

1. V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* MIT Press/Bradford Books, 1984.
2. R. A. Brooks. Intelligence without Representation. *Artificial Intelligence*, 47:139–159, 1987.
3. Laboratoire de Microinformatique l'Ecole Polytechnique Fédérale de Lausanne. Mobile Robots as Research Tools: The K-Robot Family. http://diwww.epfl.ch/Khepera/, 1998.
4. D. Floreano and F. Mondada. Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks*, 11:1461–1478, 1998.
5. N. Jakobi. Evolutionary Robotics and the Radical Envelope of Noise Hypothesis. *Journal Of Adaptive Behaviour*, 6, 1997.
6. N. Jakobi. The Minimal Simulation Approach To Evolutionary Robotics. In *Proc. of ER'98*, AI Systems Books, 1998.
7. D. Lambrinos. Navigating with an Adaptive Light Compass. In *Proc. of 3rd Eur. Conf. on Art. Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pages 602–613. Springer, 1995.
8. A. Löffler, J. Klahold, and U. Rückert. The Dynamical Nightwatch's Problem Solved by the Autonomous Micro-Robot Khepera. In *Proc. of 3rd Eur. Conf. on Art. Evolution*, volume 1363 of *Lecture Notes in Computer Science*, pages 303–313. Springer, 1998.
9. O. Michel. Khepera Simulator 2.0, Webots, 1999. http://diwww.epfl.ch/lami/team/michel/khep-sim/ and www.cyberbotics.com/.
10. O. Miglino, H. H. Lund, and S. Nolfi. Evolving mobile robots in simulated and real environments. In *Proc. of 3rd Eur. Conf. on Art. Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pages 417–434. Springer, 1995.
11. A. G. Pipe, B. Carse, T. C. Fogarty, and A. Winfied. Learning Subjective "Cognitive Maps" in the Presence of Sensory-Motor Errors. In *Proc. of 3rd Eur. Conf. on Art. Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pages 463–476. Springer, 1995.
12. E. Prem. Grounding and the Entailment Structure in Robots and Artificial Life. In *Proc. of 3rd Eur. Conf. on Art. Life*, volume 929 of *Lecture Notes in Artificial Inteligence*, pages 39–51. Springer, 1995.
13. C. Scheier and R. Pfeifer. Classification as Sensory-Motor Coordination, A Case Study on Autonomous Agents. In *Proc. of 3rd Eur. Conf. on Art. Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pages 657–667. Springer, 1995.