Evolution of a control architecture for a mobile robot

Marc Ebner

Eberhard-Karls-Universität Tübingen, Wilhelm-Schickard-Institut für Informatik
Arbeitsbereich Rechnerarchitektur, Köstlinstraße 6, 72074 Tübingen, Germany
ebner@informatik.uni-tuebingen.de

Abstract. Most work in evolutionary robotics used a neural net approach for control of a mobile robot. Genetic programming has mostly been used for computer simulations. We wanted to see if genetic programming is capable to evolve a hierarchical control architecture for simple reactive navigation on a large physical mobile robot. First, we evolved hierarchical control algorithms for a mobile robot using computer simulations. Then we repeated one of the experiments with a large physical mobile robot. The results achieved are summarized in this paper.

1 Motivation

Currently mobile robots are usually programmed by hand. For many real-world applications this introduces considerable difficulties due to the complexity of the task [10]. Instead of programming the robots directly Darwinian evolution may be used to automate this process [2]. Thus one would reduce the problem to finding an appropriate fitness function which describes how well a particular individual solves the task. The different approaches in evolutionary robotics are described by Nolfi et al. [21]. A number of researchers have already evolved control architectures for mobile robots using computer simulations. Few have worked with physical mobile robots. Brooks [4] emphasized the importance of using physical mobile robots as opposed to computer simulations. To our knowledge, those who used physical mobile robots only experimented with miniature robots such as the Khepera or a gantry-robot. Most work in evolutionary robotics was done using neural net control architectures [5, 11, 12, 9, 7, 19, 8, 18]. In contrast to this work, we wanted to see if genetic programming [13, 15] can be used to evolve a hierarchical control architecture for a simple reactive navigation task on a large physical mobile robot. If evolution is carried out on a large physical mobile robot safety measures have to be taken such that possible damage to the environment or to the robot is avoided.

As Matarić and Cliff noted [17], most research in evolutionary robotics focused on the evolution of simple tasks with long evaluation times. Recently, Nolfi [20] evolved a garbage collecting behavior for a mobile robot using accurate computer simulations of the Khepera and a neural net control architecture which also executed successfully on the real robot. Evaluation times may be reduced using computer simulations especially if more difficult tasks are addressed. Therefore it is also important to see if results from computer simulations can be transferred into the real-world.

2 Background

In this section we briefly review related approaches using genetic programming to evolve control architectures for a mobile robot.

Koza [16] used computer simulations to evolve a control program for a grid based mobile robot to mop an 8×8 area containing obstacles on some of the grids. In a more realistic setting, but also with discrete movements, Koza [14] evolved subsumption architectures [3] for wall-following for a mobile robot. Koza's representation consisted of the terminals `S00`, ..., `S11` which return distance information from twelve sonar sensors, `SS` which returns the minimum distance of all sensors, `MSD` which specifies the minimum safe distance to the wall, `EDG` which specifies the desired edging distance and the four terminals for control of the robot. The control terminal `(MF)` moves the robot 30cm forward, `(MB)` moves the robot 40cm backwards, `(TR)` turns the robot $30°$ to the right, and `(TL)` turns the robot $30°$ to the left. The evaluation was always done from exactly the same position. As primitive functions Koza used `PROGN2` which evaluates its two arguments in sequence and the conditional `IFLTE`. `IFLTE` is a four argument function. If the first argument is less than or equal to the second argument the third argument is evaluated, otherwise the fourth argument is evaluated.

Reynolds [26] used genetic programming to evolve obstacle avoidance behavior with a simulated critter. The critter moves with constant forward velocity one half of its body length per simulation step. The primitive function `turn` is used to specify the amount the critter should turn. In addition to the function `turn` the standard arithmetic functions `+`, `-`, `*`, `%` (protected division), `abs`, and the conditional `iflte` are used. The robot can perceive its environment by using the function `look-for-obstacle`, which returns a measure of the distance to an obstacle in the direction specified by its argument. As terminals he used the set of constants `0`, `0.01`, `0.1`, `0.5` and `2`. Reynolds also experimented with noise to evolve more robust controllers [27, 29] and investigated the influence the representation has on the difficulty of the problem [28]. Reynolds experimented with fixed sensors and with sensors that could be pointed dynamically. Using fixed sensors simplified the problem considerably.

Nordin and Banzhaf [22, 23, 24] used a miniature mobile robot to evolve a linear control program. Their control algorithms consisted of a variable length genome which represents an assembly program. The programs could use the following arithmetic, logical and shift operations: `ADD`, `SUB`, `MUL`, `XOR`, `OR`, `AND`, and `SHL`, `SHR`. Thus their programs are executed linearly one statement after the other. Olmer et al. [25] extended the approach by evolving control algorithms for lower level tasks and a strategy for selection of the different tasks. An overview about the research is given by Banzhaf et al. [1].

Wilson et al. [30] evolved hierarchical behaviors to locate a goal object in a maze for a mobile robot constructed from Lego Technic bricks. The primitive operations were `MoveForward`, `MoveBackward`, `TurnLeft` and `TurnRight`. Hierarchical structures are produced by chunking simple behaviors after each loop through the evolutionary process. This loop included the evolution of behaviors in simulation and then evaluating successful individuals on the real mobile robot.

In contrast to the work of Koza and Reynolds, we are working with a physical autonomous mobile robot, a Real World Interface B21. Wilson et al. and Nordin and Banzhaf also used a physical mobile robot. However our representation also includes a conditional statement. Thus a hierarchical structure emerges during the evolution. To our knowledge, so far, no one has tried to evolved a hierarchical control algorithm on a large physical mobile robot using Koza's genetic programming.

3 Representation

The representation used during the experiments described below was determined by performing a series of experiments in computer simulations. We had to chose a representation which could be used with a small population size with a reasonable number of generations [6]. Using computer simulations we found a representation suitable for the evolution of a control architecture during the allotted time.

3.1 Terminals

The terminals are used to provide sensor readings and to control the movement of the robot. All 24 sonar sensors are used to obtain information about the environment of the robot. We combined the sensors into blocks of four producing six virtual sensors as shown in figure 1. Each of the virtual sensors returns the minimum distance of the four physical sensors. The terminals for the virtual sensors are named: `FL`, `FM`, `FR`, `BL`, `BM`, `BR`.

The robot moves with a constant translational velocity of $0.1\frac{\text{m}}{\text{s}}$. Three terminals are available to the robot to control the direction of movement. The terminal `TL` starts a rotation with velocity $40\frac{\circ}{\text{s}}$ to the left, `TR` starts a rotation with velocity $40\frac{\circ}{\text{s}}$ to the right, and `RHALT` stops the rotational movement of the

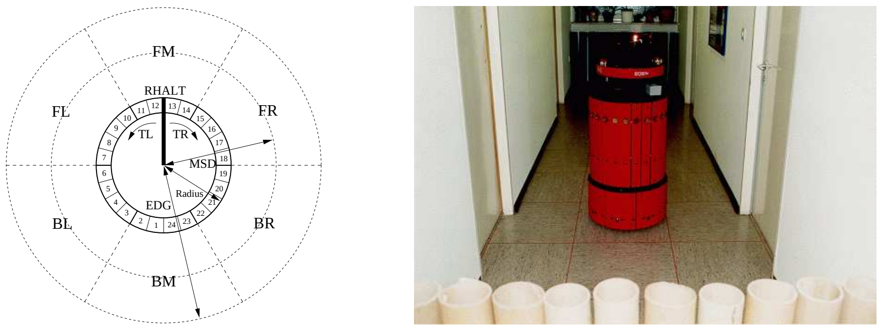

Fig. 1. Robot model seen from above with 24 sonar sensors combined to form six virtual sensors `FL` (front left), `FM` (front middle), `FR` (front right), `BL` (back left), `BM` (back middle) and `BR` (back right) which return the minimum distance of the corresponding 4 physical sonar sensors. The radius of the robot together with the minimum safe distance (`MSD`) and the desired edging distance (`EDG`) is also shown. The commands `TL` (turn left), `RHALT` (halt rotation) and `TR` (turn right) control the heading of the robot. The robot, a RWI B21, in its environment is shown on the right.

robot. In addition to executing the control command each of these three terminals also returns sensor information. The terminal `TL` returns the average value of the sensors 11 and 12, `RHALT` returns the average value of the sensors 12 and 13, and `TR` returns the average value of the sensors 13 and 14.

3.2 Primitive functions

As primitive functions we used the connective function `PROGN2` and the conditional `IFLTE`. The function `PROGN2` takes two arguments which are evaluated in sequence. The value of the last argument is returned. The function `IFLTE` takes four arguments. The first two arguments are evaluated. If the first argument is less than or equal to the second argument, then the third argument is evaluated. Otherwise the fourth argument is evaluated.

3.3 Fitness measure

In a small desktop evolution with a miniature mobile robot one may continue the evaluation of a mobile robot even if the robot hits a wall. The trajectory of the robot is simply changed due to the force exerted by the wall. With a large mobile robot such as the B21 this is no longer possible. The robot is capable of exerting a force which might cause considerable damage if the robot crashes into an obstacle. The information from the sonar sensors is monitored at all times. If an object is reported by the sonar sensors at a distance closer than 40cm from the center of the robot the motors are turned off and the evaluation is aborted. The bumpers of the robot are also monitored in case an object is not perceived by the sonar sensors. To avoid damage the next translatory movement of the robot has to be done in a direction away from the wall.

Therefore, we used a fitness measure which tries to maximize the time the robot survives until it bumps into an obstacle or its allocated time runs out. At the same time the sum of rotations should be kept at a minimum. After each evaluation the robot is repositioned using a repel operation. This operation evaluated the sonar data and the bumpers to move the robot 15cm away from the nearest obstacle. By using this operation we try to give all individuals an appropriate chance to survive. It makes no sense to start the evaluation of an individual if the robot is facing a wall in close distance.

Let T be the time available to the robot per fitness case. Let t be the time until the robot bumps into an obstacle. Let r_s be the sum of all signed rotations performed during the run and let ω be the rotational velocity of the robot. Then the raw fitness measure to be minimized is calculated according to: $\text{fitness}_{\text{raw}} = \frac{1}{n}\sum_{i=1}^{n}\left\{1 - D(1 - \sqrt{R_s})\right\}$ where n is the number of fitness cases and $D = \frac{t}{T}$ and $R_s = \frac{|r_s|}{\omega t}$. Best possible raw fitness of zero is achieved if the robot avoids obstacles while performing a balanced number of turns to the right and left. A similar fitness function was previously used by Floreano and Mondada [7, 19, 8]. The adjusted fitness to be maximized is calculated according to $\text{fitness}_{\text{adj}} = \frac{1}{1+\text{fitness}_{\text{raw}}}$. This fitness measure tries to maximize survival time while penalizing unbalanced turning.

4 Experiments

First we performed a number of experiments using computer simulations and an environment very similar to the real environment in our lab. The environment used to evolve the hierarchical control architecture can be seen in figure 1. The environment we used is rather simple because of space limitations in our lab. Occasionally people had to move through the environment but this did not prevent evolution from making progress. In other computer simulations a larger and more complex environment was used [6].

For the simulation experiments we used two different processes. One process simulates the mobile robot in its environment. The other process evaluates the control structure. Sensor information is sent from the robot process to the evaluation process and commands are sent back to the robot process. 20 experiments were performed with 1,2 and 3 fitness cases. The experiment was run for 50 generations with a population size of 75. Tournament selection with size 7 was used and the crossover, reproduction and mutation probabilities were set to 85%, 10% and 5% respectively. As maximum time per fitness case we used 300s.

In the following text we say that an individual is successful if it follows the environment without bumping into a wall. 2 successful individuals evolved with 1 fitness case, 6 with 2 fitness cases and 8 with 3 fitness cases. The performance of the individuals which evolved using 2 fitness cases is shown together with the fitness statistics in figure 2. In 50 generations with a population size of 75 a total of 3750 individuals are evaluated. We performed a control experiment with 3750 and 75000 random individuals. No successful individual was found among 3750 random individuals. The best individual found among 75000 random individuals together with a plot of the fitness values is shown in figure 2.

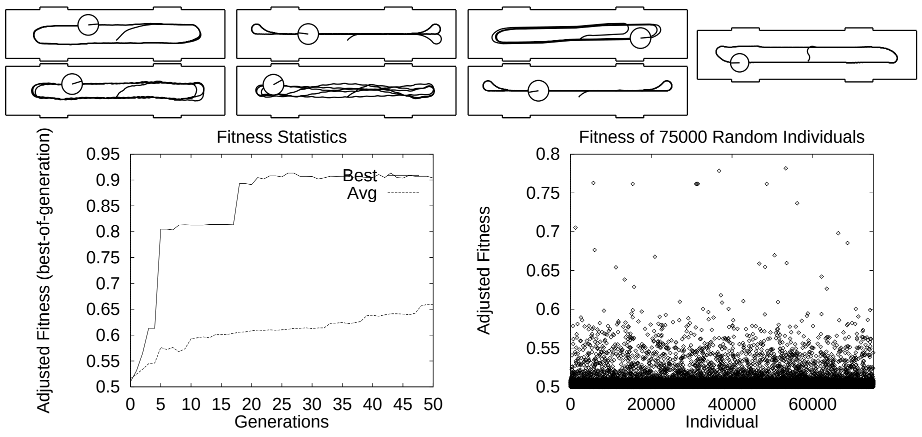

Fig. 2. 6 successful individuals evolved with 2 fitness cases. Of these the one shown on the top middle achieved an adjusted fitness of 0.9135. Fitness statistics for 2 fitness cases averaged over 20 runs and for the best run are shown on the left. The individual shown on the top right was the best individual found among 75000 random individuals. The fitness of 75000 random individuals is shown on the right for 3 fitness cases.

Next we performed the experiment using only 2 fitness cases (to limit the time needed for the evolution) on the physical mobile robot. The experiment was run for 50 generations. It took 2 months to perform the experiment on the real mobile robot. Actual time needed for the evolution was 197 hours. Initially batteries were exchanged after a generation was completely evaluated. During later generations batteries were also exchanged in between generations (just before a new individual was about to be evaluated). While the batteries were exchanged the evolutionary process was temporarily halted.

During this experiment an individual with 457 nodes reached the highest fitness. Due to the large size of the individual it is not shown here. In a series of 10 experiments where this individual was finally tested for 300s it managed to survive 8 of the 10 runs. The performance of this individual is shown in figure 3. The path was recorded using the robot's odometry and also with a standard camera using long term exposure. The average survival time of the individuals and the adjusted fitness of the best individual for each generation is shown in figure 3.

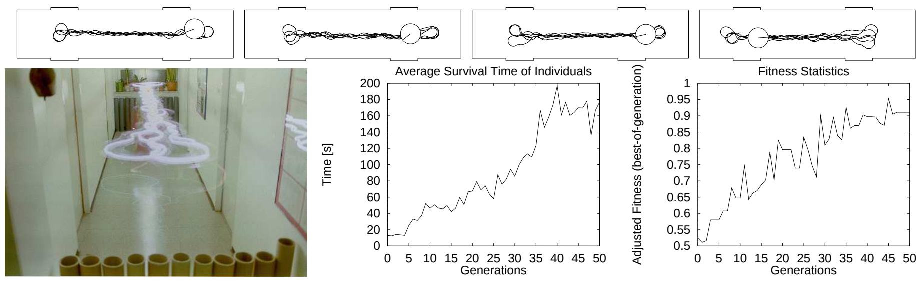

Fig. 3. Behavior of the best individual which evolved on the physical mobile robot. The paths were recorded using the robot's odometry. The photo on the left was taken at the same time the last path was recorded. The photo was taken with long term exposure while the robot was moving with a small light bulb mounted on its top. Average survival time of individuals is shown in the graph in the middle. The adjusted fitness (best-of-generation) for each generation is shown on the right.

5 Conclusion

Due to the length of the experiment we were only able to perform a single run with the real mobile robot. Although this experiment does not provide enough data to calculate reliable statistics, it shows that one is able to evolve a hierarchical control architecture using genetic programming on a large physical mobile robot. If evolution is carried out on a large physical mobile robot safety measures must be taken to avoid damage to the environment and to the robot. Although the time required for the experiment described here is prohibitive for most practical applications, one is able to use high speed computer simulations instead provided that reality is accurately modelled. On our case computer simulations were used to find a suitable representation and the main parameters for the run on the physical mobile robot.

6 Acknowledgements

The author is currently supported by a scholarship according to the Landesgraduiertenförderungsgesetz. For our experiments we used the lil-gp Programming System [31].

References

1. W. Banzhaf, P. Nordin, and M. Olmer. Generating adaptive behavior using function regression within genetic programming and a real robot. In *2nd International Conference on Genetic Programming, Stanford*, 1997.
2. V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* The MIT Press, Cambridge, Massachusetts, 1984.
3. R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, March 1986.
4. R. A. Brooks. Artifical life and real robots. In F. J. Varela and P. Bourgine, editors, *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life*, pages 3–10, Cambridge, MA, 1992. The MIT Press.
5. D. Cliff, P. Husbands, and I. Harvey. Evolving visually guided robots. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From animals to animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior, Honolulu, Hawaii, 1992*, pages 374–383. The MIT Press, 1993.
6. M. Ebner and A. Zell. Evolution of a control architecture under real world constraints using genetic programming, unpublished manuscript, 1997.
7. D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural network driven robot. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From animals to animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, Brighton, England, 1994*, pages 421–430. The MIT Press, 1994.
8. D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Trans. Syst., Man, and Cybern. B*, 26(3):396–407, 1996.
9. I. Harvey. Artificial evolution and real robots. *Artificial Life and Robotics*, 1(1):35–38, 1997.
10. I. Harvey, P. Husbands, and D. Cliff. Issues in evolutionary robotics. In J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From animals to animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior, Honolulu, Hawaii, 1992*, pages 364–373. The MIT Press, 1993.
11. I. Harvey, P. Husbands, and D. Cliff. Seeing the light: Artificial evolution, real vision. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From animals to animats 3: Proc. of the Third International Conference on Simulation of Adaptive Behavior, Brighton, England, 1994*, pages 392–401. The MIT Press, 1994.
12. I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*,20:205–224,1997.
13. J. R. Koza. *Genetic Programming, On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, Massachusetts, 1992.
14. J. R. Koza. *Evolution of Subsumption.* In J. R. Koza. *Genetic Programming I: On the Programming of Computers by Means of Natural Selection*, pages 357–393. The MIT Press, Cambridge, Massachusetts, 1992.

15. J. R. Koza. *Genetic Programming II, Automatic Discovery of Reusable Programs.* The MIT Press, Cambridge, Massachusetts, 1994.
16. John R. Koza. *Obstacle-Avoiding Robot.* In J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*, pages 365–376. The MIT Press, Cambridge, Massachusetts, 1994.
17. M. Matarić and D. Cliff. Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems*, 19:67–83, 1996.
18. L. A. Meeden. An incremental approach to developing intelligent neural network controllers for robots. *IEEE Trans. Syst., Man, and Cybern.B*, 26(3):474–485,1996.
19. F. Mondada and D. Floreano. Evolution of neural control structures: some experiments on mobile robots. *Robotics and Autonomous Systems*, 16:183–195, 1995.
20. S. Nolfi. Evolving non-trivial behaviors on real robots: A garbage collecting robot. *Robotics and Autonomous Systems*, 22:187–198, 1997.
21. S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: different approaches in evolutionary robotics. In R. A. Brooks and P. Maes, editors, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 190–197. The MIT Press, 1994.
22. P. Nordin and W. Banzhaf. Genetic programming controlling a miniature robot. In *Working Notes of the AAAI-95 Fall Symposium Series, Symposium on Genetic Programming, MIT, Cambridge, MA, 10-12 November 1995*, pages 61–67, 1995.
23. P. Nordin and W. Banzhaf. A genetic programming system learning ostacle avoiding behavior and controlling a miniature robot in real time. Technical Report SysReport 4/95, Dept. of Computer Science, University of Dortmund, Germany, 1995.
24. P. Nordin and W. Banzhaf. Real time evolution of behavior and a world model for a miniature robot using genetic programming. Technical Report SysReport 5/95, Dept. of Computer Science, University of Dortmund, Germany, 1995.
25. M. Olmer, P. Nordin, and W. Banzhaf. Evolving real-time behavioral modules for a robot with gp. In M. Jamshidi, F. Pin, and P. Danchez, editors, *Robotics and Manufacturing, Proc. 6th International Symposium on Robotics and Manufactoring (ISRAM-96), Montpellier, France*, pages 675–680, New York, 1996. Asme Press.
26. C. W. Reynolds. An evolved, vision-based model of obstacle avoidance behavior. In C. G. Langton, editor, *Proceedings of the Workshop of Artificial Life III SFI Studies in the Sciences of Complexity*, pages 327–346. Addison-Wesley, 1994.
27. C. W. Reynolds. Evolution of corridor following behavior in a noisy world. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From animals to animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, Brighton, England, 1994*, pages 402–410. The MIT Press, 1994.
28. C. W. Reynolds. The difficulty of roving eyes. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 262–267. IEEE, 1994.
29. C. W. Reynolds. Evolution of obstacle avoidance bahavior: Using noise to promote robust solutions. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming*, pages 221–241, Cambridge, Massachusetts, 1994. The MIT Press.
30. M. S. Wilson, C. M. King, and J. E. Hunt. Evolving hierarchical robot behaviours. *Robotics and Autonomous Systems*, 22:215–230, 1997.
31. D. Zongker and B. Punch. *lil-gp 1.01 User's Manual (support and enhancements Bill Rand).* Michigan State University, March 1996.

This article was processed using the LaTeX macro package with LLNCS style