ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2008, Vol.37, No.3

AUTONOMOUS MOBILE ROBOT CONTROL USING IF-THEN RULES AND GENETIC ALGORITHM

Gintautas Narvydas, Rimvydas Simutis, Vidas Raudonis

Department of Process Control, Kaunas University of Technology
Studentu St. 48-327, LT-51367 Kaunas, Lithuania

Abstract. One of the main robotics tasks is efficient autonomous mobile robots movement in previously unknown environment with obstacles and walls. Different strategies exist to design control systems to perform the robot movement. One of the simplest ways to control autonomous mobile robots is the usage of IF-THEN rules. The article shows that a rough frame of control system using IF-THEN rules can be designed. Then a genetic algorithm can by used to find optimal parameters of the control algorithm: particular boundaries of infrared proximity sensors and motors speed.

Keywords: autonomous mobile robots, IF-THEN rules, intelligent control systems, state based control systems, genetic algorithm.

Introduction

During the last decade the science of autonomous mobile robots (AMR) design has been developing very fast [2, 4, 10]. Evolutionary robot techniques and evolutionary programming methods provide new opportunities for evolutionary AMR control systems design [5, 6]. Stationary, wheeling, crawling, creeping, walking, swimming, and flying robots are designed. They are able to see, hear, communicate among themselves, and even talk to people. Moreover, they are able to plan their actions. Robots are able to wash rooms, launder, make food, search for certain things, transport them to specific places, orient in stable and shifting environments, interact with them, help men control sophisticated technologies, perform dangerous tasks, and perform tasks in an environment where men cannot get into or stay within. They can serve as guards of a special territory or simply to be sources of intelligent entertainment.

However AMR cannot move, communicate, recognize and orient in the environment as well as the life creatures can. The intelligent control systems are neither universal nor powerful enough to guarantee a good working of AMR. In the robotics it is necessary to improve both the hardware and the software, to pay exclusive attention to the intelligent hybrid control systems and evolutionary algorithms. In order to design the effective control systems we have to implement two stages:

1) to create a hybrid intelligent control system for solving a specific task;

2) to use evolutionary programming methods in order to improve the results of the first stage.

In this article both stages are described. The first stage – how to create a frame of a state based control system, and the second stage – methodology how to improve control system using genetic algorithm.

In the experiments we have taught the AMR Khepera II to follow the walls of the maze using control system made of IF-THEN rules. The ability to follow the walls can be easy transformed into the ability to avoid even moving obstacles and to circuit them on the right or on the left. It is shown that the IF-THEN rules can be used as the control systems for AMR. One input system with seven IF-THEN rules was applied. During the experiment it was sought optimal infrared sensors boundaries, which give the best result for the wall following task.

1. Robot Khepera II

All experiments were made using the miniature mobile robot Khepera II [7]. The robot and its view from the top are shown in Figure 1.

The robot's diameter is 70 mm, height 30 mm, weight about 80 g. It is designed to move on the flat surface. It has two lateral wheels that can rotate in both directions in appointed speed. Under the robot two rigid pivots in the front and in the back are installed. They do not let the robot sway forward and backward.

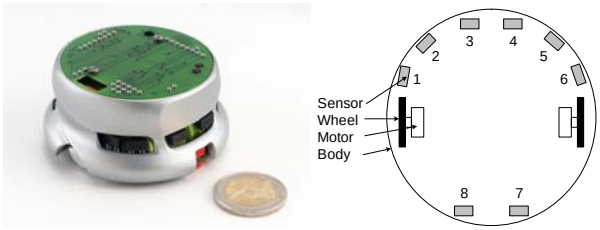
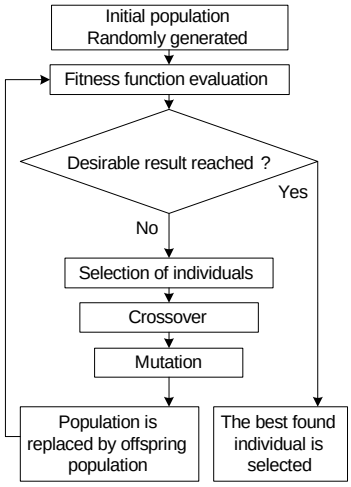


Figure 1. The miniature mobile robot Khepera II and its view from the top [7]

Various additional turrets can be added on the top of the robot basis: linear vision module, gripper, vision turret, and communication turrets etc. Eight infrared proximity sensors are located around the body: six sensors in front and two sensors on the back side. Using Khepera's sensors it is possible to detect obstacles and to estimate distances up to them. Values of each sensor are integer numbers from 0 to 1023. The higher the value of the sensor, the nearer the obstacle is. The maximum sensitivity distance is about 100mm. The robot can be attached to the computer through a serial cable and RS232 connector, which can be used to transfer data and to supply power. Miniature size of the robot gives few advantages using it in experiments. It is possible to build various complex environments directly on the table, for example mazes. A little working surface lets us keep a close watch on the behavior of the robot during experiments. Ones of the most important advantages of the miniature robot are its size, weight, and ability to move fast. Therefore, when the robot hits a wall or an obstacle, it is neither damaged nor broken. Application possibilities of the robot are wide. It can be used to design and to develop the movement, recognition, orientation in an environment, path planning, control, cooperation, and other algorithms.

2. Genetic Algorithm

Genetic algorithm (GA) [3, 8] is a useful instrument of evolutionary computation for optimization. In general, it is a search algorithm, which uses principles inspired by the Nature. GA differs from a direct search procedure. This method suits when we do not need to find the best solution and it is enough to find near solution to the best one from a huge space of possible solutions. The major weakness of the GA is that it usually tends to be computationally expensive in the real systems. Evaluation of fitness function is an expensive process and takes much time. In Figure 2 the scheme of GA is presented.



Crossover with a single crossover point
0101|10100 ⇒ 010111001
1100|11001 ⇒ 110010100

Mutation
110010100 ⇒ 110011100

Figure 2. Scheme of Genetic Algorithm

GA operates with randomly generated population of individuals encoded by binary strings (chromosomes) which correspond to particular solutions. This population is a small part of all possible solutions. Population evolves toward better solutions while genetic operators such as selective reproduction, mutation, and crossover are applied. In every generation, better individuals (solutions) generate offspring which inherit better characteristics and replace worse individuals during generations. The fitness function is a performance criterion that evaluates the performance of each individual. Individuals with higher fitness values are better. Selective reproduction is based on natural selection. Members of the population are chosen for reproduction on the basis of their fitness defined according to some specified criteria. The best individuals are given a greater probability of reproducing in proportion to the value of their fitness. There are a few basic methods to implement selection: roulette wheel, truncation selection, selection based on tournament. Often it is useful to preserve the best individual(s) for the next generation. This strategy is

known as elitism. Crossover lets two members of the population exchange genes. There are many ways to implement crossover. It is possible to have a single crossover point or many crossover points. These crossover points are selected randomly. The last procedure of GA generation is mutation. Here a randomly selected particular gene in a particular chromosome (solution) is changed. Thus a 0 is changed to a 1 or vice versa. The process of mutation is performed rarely.

3. Description of the experiments

The aim of the evolutionary experiments was to create an intelligent control system for AMR using IF-THEN rules and GA that would allow AMR to follow the wall of the maze bearing right hand rule. For the experiment, i.e. for the evolution of the control system we used a maze 60x40 cm shown in Figure 3. After evolution process, the robot must be able to circuit the maze following the wall on the right side.

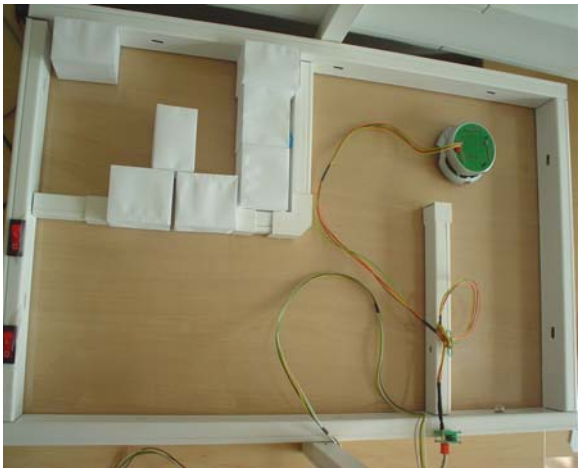


Figure 3. The maze used in the experiments

To evaluate control system's ability to control the robot, the following fitness function is used:

$$fitness = (0.05\,a + 0.05\,b + 0.2\,c + 0.7\,d)\cdot\left(\frac{t}{T}\right)^2 \rightarrow \max, \quad (1)$$

$$a = 1 - \frac{1}{40N}\sum_{i=1}^{N}\left|S_{Li} - S_{Ri}\right|, \quad (2)$$

$$b = 1 - \frac{1}{40N}\sum_{i=1}^{N}\left|\Delta_{Li}\right| + \left|\Delta_{Ri}\right|, \quad (3)$$

$$c = 1 - \frac{1}{N(1023 - A)}\sum_{i=1}^{N}\left|S_{5i} - A\right|, \quad (4)$$

$$d = \frac{1}{40N}\sum_{i=1}^{N} S_{Li} + S_{Ri}\,. \quad (5)$$

Here N is the number of measurements made during experiment every 0.07 second. S_L and S_R are speeds of the left and right wheels. Δ_L and Δ_R are the deviations of the current wheel speeds from the regulation speeds. These deviations emerge due acceleration and inertia. S_5 is a current value of the fifth infrared sensor (Figure 1, right). A is a constant which shows desirable distance of the robot from the wall or obstacle. Therefore a is a measure of snaking, b is a measure of twitch, c is a measure of the deviation from the regulation distance from the walls, and d is a measure of average speed. t is a time which robot runs controlled by control system. T is a maximum possible time of the run while control system with one collection of the parameters controls the robot. In our experiments, T is 40 seconds. If the robot can follow the wall, it runs all 40 seconds and t is 40 too. But if the robot can not follow the wall: it turns from the wall, its wheels speed is very slow or negative or it sticks, it is stopped earlier ant then t is less than 40. The ratio t/T is squared that we could better separate individuals with better skills. Architecture of the robot lets us get all the data for the calculating of these measures. It is evident that the robot must snake and twitch less, keep constant distance from the wall on the right side, and move as fast as it is possible following the wall. Since the speed and distance from the wall are the most important, these two components have bigger weight.

Our created intelligent control systems are state based control systems. They are similar, shown in Figure 4. Each system has seven states. They have only one input of the fifth sensor (Figure 1, right). The difference between these systems is that the first one (Figure 4, top) has more degrees of freedom choosing the values of the wheels speed. In the second system (Figure 4, bottom), in a few cases speed is set to maximum.

If the robot is far from the wall, the minimal infrared sensors value is 80. If the robot touches a wall, the maximum sensors value is 1023 (Figure 4). Due to the similarity of the systems, we will describe only the first one.

State$_1$ – robot is too far from the wall or obstacle and can not see them. Robot moves directly forward in maximum speed while does not see a wall. The speed of the left wheel S_L is equal to the speed of the right wheel S_R.

State$_2$ – robot is far from a wall but can feel it. Khepera does rude turn towards the wall. S_L is lot bigger than S_R but S_R is positive.

State$_3$ – robot is not far from the wall, but it is further than the optimal distance (in our task – approximately 2 cm). S_L is near maximum and S_R is a little bit less. The robot little by little approaches the wall while it is following it.

State$_4$ – robot is in a good range from the wall and goes forward.

State$_5$ – robot is not far from the wall, but it is closer to the wall than the optimal distance. S_R is near maximum and S_L is a little bit less. The robot little by little recedes from the wall.

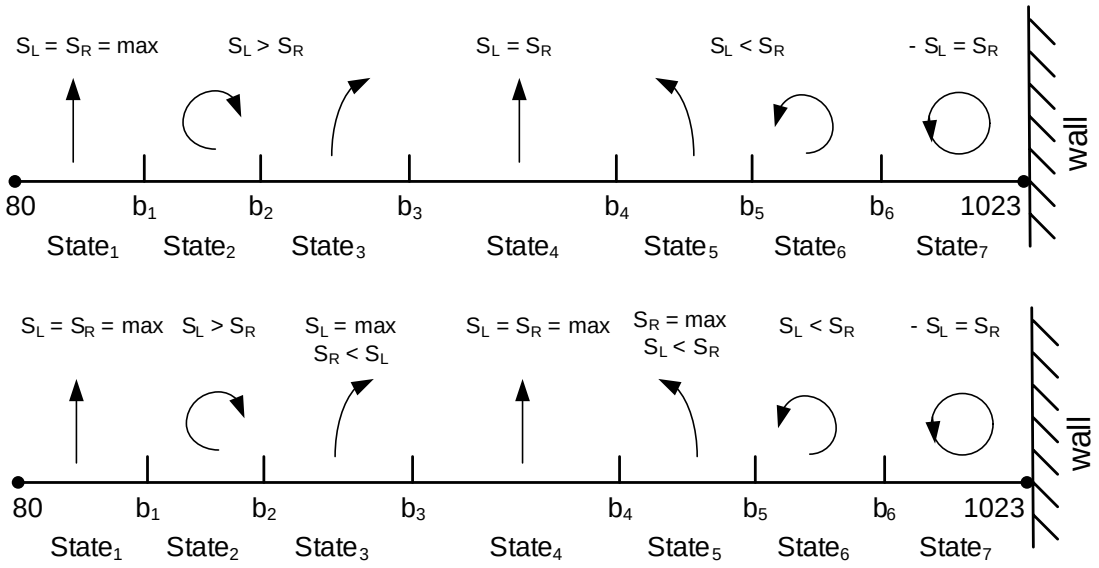


Figure 4. Schemes of intelligent state based control systems

$State_6$ – robot is too close to the wall. Khepera does rude turn from the wall. S_R is lot bigger than S_L but S_L is positive.

$State_7$ – robot almost touch the wall. It starts to turn on its axis anticlockwise. The speed of the wheels is equal in absolute magnitude but have opposite sign. The robot does not move, only turns on its axis.

$State_1$ and $State_7$ do not control the robot while it follows the wall. These states let the control system prepare the robot to follow the wall. Control system has to control in such way that the robot would never hit these states.

GA with population of 50 individuals was used. In the first control system, each individual is encoded in binary chromosome with 98 genes. 44 genes are responsible for an optimal speed and others 54 are responsible for the optimal boundaries b_1, b_2… b_6. In the second control system, each individual is encoded in binary chromosome with 86 genes. 32 genes are responsible for an optimal speed and others 54 are responsible for the optimal boundaries. Five best individuals from the population were preselected in each GA generation. Other individuals were selected using roulette wheel principle. Crossover was implemented using a single randomly selected crossover point. Mutation usually occurred with the declining probability during the generations from 0.04 till 0.005.

4. Results

The main measure – *fitness* that shows quality of the robot control was observed. In Figure 5 we can see average fitness of the population of 50 individuals and the fitness of the best individual from each generation. The first system is shown on the left and the second system on the right.

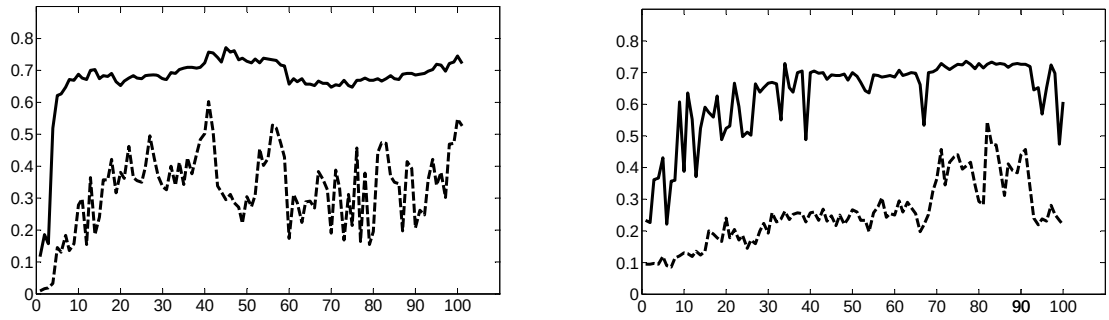


Figure 5. Fitness functions of the best individual and average fitness of the population during GA generations

The system with more degrees of freedom for the speed values search showed better results. It needed less generations to find better individuals. The second control system where some speed positions were set to the maximum speed (Figure 4, bottom), needed longer time to find good solutions which have fitness function values more than 0.7. If we compared average populations fitness of the systems, we would see that the first system tends to have better population. The curve of the best individuals in the first system

showed more stability but the best results of both systems are similar.

5. Conclusions and recommendations

In this paper the usage of IF-THEN rules and Genetic Algorithm for design of intelligent control systems for AMR is presented. We showed that the framework of the control system can be designed and then Evolutionary Algorithms can be used to find optimal parameters of the control system. The most difficult problem in our task is to find such parameters which could match up fast speed and precise movement. When the speed grows up, the robot can follow a wall but it can not react in time to the wall ahead. If the robot had to follow straight wall, it would be a simple task and it would be enough five states. But when we have environment with inner and outer corners, the task becomes quite complicated. As our experiment showed, the maximum speed is not the best solution. The part of the experiment which took the most of time was the robot Khepera II movement and evaluation of the fitness function. The ability to follow walls can be easy transformed into the ability to avoid even moving obstacles and to circuit them on the right or on the left. We measured fitness when the robot Khepera was controlled by human expert operator via joystick. He reached 0.711 fitness value [9]. This is a similar result to our gained, but after 15 minutes of control the expert operator gets tired, loses concentration and can not control the robot very well.

We should try to design intelligent control system for AMR using IF-THEN rules, FUZZY logic, and GA. It should be better because of absence of the gaps of the wheels speed which occur during the jumping among the states. FUZZY logic will let change speed of the wheels continuously.

Our designed control systems can be a part of Behavior-Based control systems [1].

References

[1] **R.C. Arkin.** Behaviour-Based Robotics. *Cambridge, The MIT Press*, 1998.

[2] **G.A. Bekey.** Autonomous Robots: From Biological Inspiration to Implementation and Control. *The MIT Press*, 2005.

[3] **J.H. Holland.** Adaptation in Natural and Artificial Systems. *The MIT press*, 1992.

[4] **J.M. Holland.** Designing autonomous mobile robots: Inside the mind of an intelligent machine. *Elsevier*, 2004.

[5] **D. Floreano, F. Mondada.** Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 1996, 26(3), 396-407.

[6] **D.B. Fogel.** Evolutionary computation: Toward a New Philosophy of Machine Intelligence. 2nd *edition, Piscataway, NJ: IEEE Press*, 1999.

[7] **K-Team.** Khepera II User Manual. *EPFL, Lausanne*, 2002, *http://ftp.kteam.com/khepera/documentation/Kh2UserManual.pdf.*

[8] **Z. Michalewicz.** Genetic Algorithms + Data Structures = Evolution Programs. *Berlin: Springer*, 1996.

[9] **G. Narvydas, R. Simutis, V.Raudonis.** Learning action of skilled operator for autonomous mobile robot control. *Proceedings of International Conference "Electrical and Control Technologies – 2007", Kaunas*, 2007, 78-81.

[10] **S.Nolfi, D.Floreano.** Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines. *The MIT Press*, 2000.

Received April 2008.