Real-Time Genetic Obstacle Avoidance Controller for a Differential Wheeled Exploratory Robot

Adriana Sîrbu[1] and Dan-Marius Dobrea[1]
[1] Technical University "Gh. Asachi", Faculty of Electronics, Telecommunications and Information Technology, Iaşi, Romania
mdobrea@etti.tuiasi.ro , asirbu@etti.tuiasi.ro

***Abstract*— This paper presents the development of an autonomous differential wheeled robot able to avoid short-distance obstacles using signals from a set of infrared sensors. For the proposed implementation, the situations of imminent collision are solved on line using an adequately designed genetic algorithm (GA) . In this way a knowledge database comprising the main set of rules that directly map the sensor information into the engine commands can be developed on the fly. Our experiments proved that a satisfactory behavior can be obtained even in cases when no initial knowledge database, usually obtained previously off-line through simulations, is provided. The implementation uses a MFC5213 Freescale microcontroller. The GA is developed in C language, using the CodeWarrior 7.2.2 IDE and was extensively tested to prove the viability of the proposed solution.**

I. INTRODUCTION

Recently, due to the progress of the electronics industry that provides very powerful processors (operating at higher processing frequencies, having multiples cores, larger memory, larger word size etc.) sophisticated algorithms can be implemented in embedded systems. From this perspective, the soft computing techniques (Artificial Neural Networks (ANN), Fuzzy Logics (FL), Evolutionary Computation (EC)) received a particular interest especially in the field of the autonomous robots. In such applications, the controller adjusts autonomously the motor command whenever the robot is in the imminent danger of colliding. These techniques are extremely adequate for processing real world information, involving tolerance of imprecision and uncertainty. As a result, real-time robot control applications are possible based on all the above mentioned soft computing techniques.

The fuzzy logic approach was one of the first choices in this context. The fuzzy logic controllers, which use type-1 fuzzy sets, are widely used. The fuzzy logic controllers, which use type-1 fuzzy sets, are widely use [1]. But the dynamic unstructured environments generate problems regarding the antecedents' and consequents' membership function design, resulting sub-optimal type-1 fuzzy sets for specific environments and operation condition [1]. As a result, more recently, type-2 fuzzy sets start to be used in mobile robots control and in industrial applications [2], [3].

Expressing a problem with uncertainties as a stochastic programming application ensures the ability of a GA to solve it [4]. Consequently, the GA controller is able to deal with all the existing problems of a real-time robot control.

In the literature several hybrid controllers for robots – GA-fuzzy or GA-neural, are described. In these approaches, the GA adjusts the parameters of the controller, obtained by means of an evolutionary process, providing so different behaviors of the robot. During the evolution, the controllers with the best performances have more chances to spread their characteristics to the offspring and so, after several generations, a robot controller with better performances is obtained. There are several successful reported applications in which the GA tunes the fuzzy logic controllers [5-7]. Ant colony algorithm is also used to enhance the GA in the design of the fuzzy controllers devised for obstacle avoidance, [8].

One of the main applications of the GA in robotics consists in finding the optimal path that should be followed by a robotic system in order to reach a particular destination [9], [10]. The main stages of designing a reliable navigation algorithm for an autonomous robot are: localization of the current position, autonomous execution of a local collision-free motion within its environment and, eventually, finding out a global optimal path to its final destination.

For most of the implementations of the mobile robot navigation, the support of its "intelligence" is offered by a previously constructed knowledge database which holds the solutions of avoiding obstacles for some particular environments, practically a set of rules providing adequate motor commands. In order to obtain these databases in FL the knowledge of human experts is used [11].

The work presented in this paper represent a continuation of that one presented in [12] where the core movement knowledge-database was determined off-line using a simulation environment for robotic systems in which a GA is used to extract the above mentioned rules. Unlike the previously mentioned approach, in this paper the rules of the knowledge database are extracted using a GA that runs in real time directly on the robot.

There are numerous critical applications, like space missions, where the robot's mission could be compromised due to a failure and so, a large quantity of resources and efforts could be lost forever. To prevent a failure of the controller involved in the obstacle avoidance behavior (in which the core movement knowledge-database was previously obtained) a self-organized intelligent robotic controller must be used to obtain in real time a new movement knowledge-database in such situations.

The new proposed implementation solves the problem of devising a real-time local collision-free robot trajectory

extracting a new knowledge-database in the case of a malfunction of the original movement knowledge-database.

The results obtained in a previous paper [12] and the results obtained in this paper confirm the validity of the GA approach for obstacle avoidance problem.

The next two sections present the robot hardware and software architectures. Experimental results are commented in Section IV, followed by conclusions.

II. HARDWARE ARCHIECTURE OF THE ROBOT

The designed robot has three degrees of freedom, being able to execute two basic movements: rotation and translation. In the rear part of the robot there two motor-driven wheels are placed providing locomotion through differential drive mechanism, see Figure 1. The two direct current engines that drive the wheels are, in turn, controlled by a microcontroller system. An unpowered wheel, placed in the frontal-central part of the robot, ensures stability. The robot has an average top speed of 0.3 m/s.

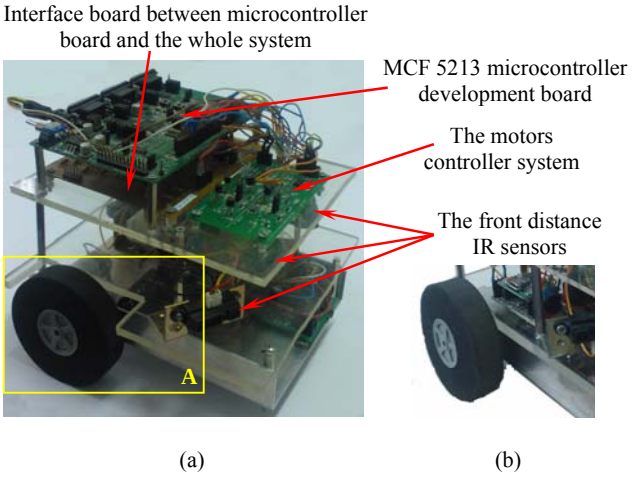

Figure 1. (a). Robot prototype; (b) detail of the previous implementation [12]

In the new implementation of the robotic system several improvements have been realized. One of them concerns the mechanical structure of the robot. The wheels are now placed inside the structure of the robot, as shown in Fig. 1(a), as compared with the older version, see details in Fig. 1(b) [12]. In this way, the left and right sensors protect the wheels from crashing into obstacles from the environment and, much more, the learning procedure proved to be significantly improved.

The robotic system is equipped with four infrared, IR, proximity sensors – GP2D120XJ00F, with an active distance measuring range from 4 up to 30 cm. Three of the IR sensors are placed in front of the robot, supervising the left, the central and the right part of the frontal environment and one is placed in the central-rear position. The sensors provide voltages proportional with the distance to the detected obstacles. The sensor characteristic was linearized based on a regression algorithm [13].

The GA based controller is built on a powerful 32-bit MCF5213, a Freescale RISC architecture microcontroller, [14]. It belongs to the ColdFire™ family having a large number of peripheral equipments such as eight PWM channels, four 32-bit timers with DMA request capability, eight channels ADC, 3 UARTs, 1 CAN.

The speed of the wheels is updated at 3.3 Hz, through two PWM channels that command two H-bridges.

III. SOFTWARE ARCHTECTURE

The software architecture can be divided into two different layers. The top level is a responsible for the data acquisition and navigation functionalities. The lower level controls the motor. Several special functions were designed in assembly language and in C to solve the appropriate exploratory motion requirements and to control the whole robotic system.

The main characteristic of an autonomous robot is its ability to learn, adapt and generalize the knowledge related to its interaction with the environment.

The use of GA for solving the problem of collision-free exploratory motion represents a new approach, [12]. The learning and adaptation processes are based on the genetic evolution of an adequately designed population of individuals whose performances are evaluated by a special fitness function.

A. *Genetic Algorithms*

A genetic algorithm is a heuristic search procedure inspired from the natural process of evolution as in biological sciences. The algorithm operates over a population of possible candidate solutions, called chromosomes or individuals [15].

Each chromosome is evaluated and ranked by means of a fitness evaluation function. The fitness function provides information of the adequacy of each individual. The evolution of the GA from a generation to the next one involves the following steps: fitness evaluation, chromosome selection, crossover, mutation and building the next generation. The population genetics evolves in a given environment according to the natural behavior in which the fittest survive and the weakest is destroyed and so the next generation is created with the only goal of improving the population fitness. Different stopping criteria can be introduced in order to complete de evolution process.

The general description of the GA is given in Fig.2 .

All selection methods are based on the same principle: giving fitter chromosomes a larger probability of selection. Common methods for selection are: Roulette Wheel selection, Stochastic Universal Sampling, Tournament selection.

New individuals, offspring, are generally created as offspring of two parents by selecting (usually at random) one or more crossover points within the chromosome of each parent. The parts delimited by the crossover points are then interchanged between the parents.

New individuals are also created by making modifications to one selected individual by means of so called mutations. The modifications can consist of changing one or more values in the representation.

```
Initialize
        Pop             // Randomly created population
        Cross_Prob      // Crossover probability
        Mut_Prob        // Mutation probability
        Max_Gen         // Maximum no. of generations

Procedure
        Gen=1
        Compute_fitness(Pop)
        While(Gen<=Max_Gen)
                Sel_pop = Select(Pop)
                Pop = Crossover(Sel_pop, Cross_Prob)
                Pop = Create_NewPop(Sel_pop, Pop)
                Pop = Mutate(Pop, Mut_prob)
                Compute_fitness(Pop)
                Gen=Gen+1
        EndWhile
        Optimal=Min_fitness(Pop)
EndProcedure
```

Figure 2. GA Algorithm

B. GA Implementation

In our implementation, each individual of the GA population has two genes of 8 bits each: the left and right engine commands respectively. As a result, each chromosome is represented on 16 bits.

Two different representations of the genes have been investigated for this GA implementation: a binary and a floating point one [16]. Finally we have adopted the real valued representation, which gave us the possibility to adjust the precision of the representation and so, to control the size of data segment as well as the execution time.

Special care had to be taken in order to optimize the length of the code so as to fit the flash memory of the controller. A difficult task was also the minimization of the length of the data segment so as to fit the RAM capacity. We have designed special data structures (unions to overlap buffers of data) while using the possible minimum length of the variable representation (*short int* instead of *int*, *float* instead of *double*)

After several tests and taking into account the restrictions imposed both by the size of the RAM and the real-time execution, we have adopted a population with ten individuals. The designed GA has the following parameters: crossover probability *Cross_Prob* =0.7, mutation probability was chosen in connection with the length of the chromosomes (Lind), which depends on the imposed representation precision, *Mut_prob* =0.7/Lind [16], generation gap=0.9, maximum number of generations *Max_Gen* =20.

We have implemented the one point crossover operator and the Stochastic Universal Sampling selection method.

The analytic form of the fitness function, $f(\cdot)$, is presented in equation (1). The relation (1) complies with the fundamental paradigm that defines the fitness in the genetic algorithm field: the fitness takes the lowest value (zero in our situation) only when a chromosome successfully solves the problem. In other cases, the fitness measures the ability of each chromosome to solve more or less the problem – being in a direct proportional relationship with the vicinity of an obstacle, as it is our case.

$$f_j\left(c_j[n]\right) = \frac{1}{4}\sum_{i=1}^{4} s_i[n] \tag{1}$$

where n represents the current generation.

After 300 ms of robot movement, based on the $c_j[n]$ chromosome containing the engine commands, the fitness is calculated. A collision-free motion, specified by a chromosome $c_j[n]$, is characterized by values of 0 for all four sensors (this corresponds to the "no obstacle" case, in the vicinity of the robot); in such a situation the fitness function, $f(c_j[n])$, is also zero.

With the aim of speeding up the GA execution we have stopped the genetic evolution if the fitness function of the best individual is less than 0.1.

In order to obtain a set of adaptive movement rules – based on a continue interaction of the robot with the environment and without any human intervention, in a first main stage the robot has to go straightforward. When the robot comes close enough to an obstacle (i.e. the obstacle lies in the active range of its sensors), the GA starts to find the best solution in order to avoid the obstacle; this last case corresponds to the second main stage of the algorithm. In this last stage, the most important goal of the real-coded genetic algorithm is to find the best chromosome(s) that encapsulates those engine commands that minimize the fitness function. As a result, the GA finds the best way to avoid the obstacle.

The GA works with population of chromosomes. In other researches, each chromosome characterizes a single robot [7]. In our case, we have only one robot that operates the GA, which can be considered a major practical limitation. To avoid this limitation, in a first step, the robot will move in one direction based on the engine commands encapsulated in the first chromosome from the population. After 300 ms the robot stops and the fitness value associated with the chromosome is determined. Then, the robot comes back into the initial position and the algorithm proceeds, in the same way for all the other chromosomes.

Finally, the behavior of the robot can be considered as similar to that of a human that reacts and learns through experience.

IV. Experimental Results

The proposed implementation was tested in a real environment with static obstacles (in both learning and avoiding phases). The GA self-organized intelligent controller runs on the robotic system and interacts with the environment on-line and in a real-time fashion. The interaction with the environment is made based on its sensors and the engines (effectors) and the result of these interactions is a set of rules (extracted based on an auto-organizing process).

The execution time for evaluating a generation of the GA is 11 seconds.

In Table I we present some of the results obtained during the experiments: in the first four columns – the information

acquired from the IR sensors (0 – no obstacle in the sensor range, 1 – imminent collision); the last columns – the engine commands as resulted from the GA.

TABLE I. MOVEMENT KNOWLEDGE DATABASE – REAL TIME RECORDINGS

Sensors				Engines	
Left	*Center*	*Right*	*Back*	*Left*	*Right*
⋮	⋮	⋮	⋮	⋮	⋮
0.3669	0.3133	0.1514	0.0	-0.4117	-0.1607
0.3278	0.3325	0.3669	0.0	-0.6235	-0.1529
0.0	0.4306	0.8012	0.0	0.4376	0.7242
0.7229	0.3301	0.0	0.0	0.4509	0.2
0.5326	0.4278	0.3472	0.0	0.6741	0.2302
⋮	⋮	⋮	⋮	⋮	⋮

We have thoroughly investigated the robot behavior in different situations as function of the relative position with respect to the obstacles inspecting the created knowledge database. One interesting particular case is illustrated in Fig. 3.

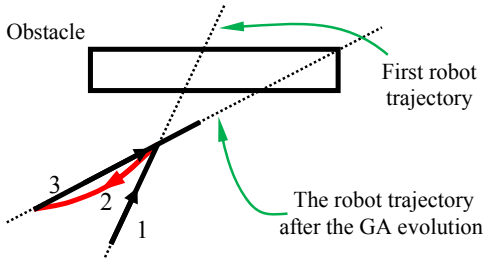

Figure 3. Obstacle avoidance case

What could appear as a trivial solution, for example in the cases when the final decision at a certain moment is moving backwards (negative commands for the engines), depicted as trajectory 2 in Fig. 3, proved to be a feasible one as it places the robot in a position which is more appropriate for avoiding the obstacle (the next decisions for engine commands will effectively avoid it).

V. CONCLUSIONS

This paper presents the implementation of an autonomous wheeled robot using a GA based intelligent strategy to avoid obstacles. The novelty of the proposed solution consists in the absence of an initial knowledge database. The robot is able to determine the trajectory which allows the avoidance of the obstacle, to store the solution (engine commands correlated to the signals acquired from the sensors) and to create in real-time a knowledge database for further environment exploration. The practical implementation using MCF5213 Freescale family microcontroller validated the proposed solution.

ACKNOWLEDGMENT

The authors are grateful to the Silica (an Avnet Company) for the generous donation of a set of Freescale development boards.

REFERENCES

[1] H. Hagras, C. Wagner, "Towards the Wide Spread Use of Type-2 Fuzzy Logic Systems in Real World Applications," IEEE Computational Intelligence Magazine, vol. 7, no. 3, 2012, pp. 14 – 24.

[2] H. Hagras, "Type-2 FLCs: A new generation of fuzzy controllers," IEEE Comput. Intell. Mag., vol. 2, no. 1, Feb. 2007, pp. 30–43.

[3] T. Dereli, A. Baykasoglu, K. Altun, A. Durmusoglu, and I. Burkhan Türksen, "Industrial applications of type-2 fuzzy sets and systems: A concise review," Comput. Ind., vol. 62, 2011, pp. 125–137.

[4] Yasunari Yoshitomi Hiroko Ikenoue Toshifumi Takeba Shigeyuki Tomita, "Genetic Algorithm in Uncertain Environments for Solving Stochastic Programming Problem," Journal of the Operations Research, Vol. 43, No. 2, June 2000, pp. 266-290.

[5] S. Tan, S.X. Yang and A.M. Zhu, "A novel ga-based fuzzy controller for mobile robots on dynamic environments with moving obstacles," International Journal of Robotics & Automation, Vol. 26, Issue 2, 2011, pp. 212-228.

[6] R. Martíneza, O. Castilloa and L.T. Aguilarb. "Optimization of interval type-2 fuzzy logic controllers for a perturbed autonomous wheeled mobile robot using genetic algorithms," Information Sciences, Vol. 179, Issue 13, 2009, pp. 2158-2174.

[7] C. Messom, "Genetic algorithms for auto-tuning mobile robot motion control," Research Letters in the Information and Mathematical Sciences, Vol. 3, 2002, pp. 129-134.

[8] J.S. Chiou, C.J Wang, K.Y. Wang, Y.C. Hu, S.W. Cheng and C.H. Chen, "Hybrid algorithm of FLC design for robot soccer," International Journal of Nonlinear Sciences and Numerical Simulation, Vol. 11, 2010, pp. 119-122.

[9] A. Hosseinzadeh and H. Izadkhah, "Evolutionary approach for mobile robot path planning in complex environment," International Journal of Computer Science Issues, Vol. 7, Issue 4, No 8, 2010, pp. 1-9.

[10] P.P. Repoussis, C.D. Tarantilis and G. Ioannou, "Arc-guided evolutionary algorithm for the vehicle routing problem with time windows.", IEEE Transactions on Evolutionary Computation, Vol. 13, Issue 3 (June), 2009, pp. 624 - 647.

[11] Tan A.H., Lu N. and D. Xiao. "Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evalutive feedback," IEEE Transaction on Neural Networks, Vol. 19, 2008, pp, 230-244.

[12] D. M. Dobrea, A. Sirbu, M.C. Dobrea, "A self-evolving controller for a physical robot: a new introduced avoiding algorithm 12th Middle Eastern Simulation and Modelling Conference (MESM) and 2nd GAMEON-Arabia Conference (Edited by Marwan Al-Akaidi and Ken Newman, Publication of Eurosis-ETI, Printed in Ghent, Belgium), November 14-16, 2011, Arab Open University, Amman, Jordan, pp. 65-70.

[13] D.M. Dobrea and M.C. Dobrea, "An auto-organization bio-inspired robotic system", International Conference on Future Information Technology, ICFIT 2010, December 14-15, 2010, Changsha, China, vol. 2, pp. 354-358, ISBN: 978-1-4244-8370-9, IEEE Catalog Number: CFP1088K-PRT.

[14] http://www.freescale.com/

[15] Z. Michalewicz, Zbigniew, "Genetic Algorithms + Data Structures = Evolution Programs," Springer-Verlag, 1999.

[16] C.Z. Jaiiikow and Z. Michalewicz, "An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms", Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann Publishers, 1991.

[17] http://www.geatbx.com/docu/index.html