Simple Target Seek Based on Behavior

LUBNEN NAME MOUSSI and MARCONI KOLM MADRID
DSCE – FEEC
UNICAMP
Av Albert Einstein, 400 – Cidade Universitária Zeferino Vaz, Campinas, SP
BRAZIL

Abstract: - This paper shows that it's possible to design and implement a very simple architecture that can solve the classical problem in mobile robotics control, which is *target seek while performing safe exploration of the environment*. The authors, interested in autonomous mobile robotics, tell their story designing it starting with ideas from possible solutions given by Evolutionary Robotics and ending with a simpler choice utilizing Behavior-Based Robotics. The solution was implemented in simulation, considering real robots characteristics and inside office environment, to demonstrate its feasibility and to produce more realistic results, with greater chances of application in the real world.

Keywords: - autonomous mobile robotics, evolutionary mobile robotics, behavior-based robotics, path determination, target seek.

1 Introduction

The authors, interested in autonomous mobile robotics, started their work implementing obstacle avoidance, as can be seeing in [16], [17] and [15], utilizing classifier systems [3], [8] and introducing the neural network classifier system. Classifier systems have learning ability in generating valid rules by its own, without the need of a designer formulating them, but feeding the classifier system with some valid rules to start with makes the learning process faster. Learning has the additional advantage of permitting the acquisition of new rules when the environment changes. This work required great effort to design and implement.

Solved obstacle avoidance, it was time to solve target seek. The first idea was to utilize the architecture developed for obstacle avoidance and integrate it with target seek. Looking for related work it became clear that it was not an easy task, as can be seeing in [19], with samples that one can try through a free simulator [18]. A work in target capture with collision avoidance, using a modified version of classifier systems is found in [7], were it is clear the difficulties and complexity in design, besides the simplicity of the environment. In [14] it is presented the challenges of evolving controllers for physical robots.

Looking for an easier way to implement the intelligent control system and having some introductory notions of the relevance of the work done by Brooks in Behavior-Based Robotics, the authors decided to get in touch with it. It rapidly revealed to be a radically different approach, as can be seeing in the work developed by Maja Mataric [10], related to mobile robotics and a complete book in the subject written by Arkin [1].

Behavior-Based Robotics (BBR) looked simpler and the authors decided to start with it in a way to take advantage of the experience already acquired with classifier systems. For doing that each behavior would be developed in two stages: the first based in BBR that would provide a minimum set of good rules that were to be evolved in the second stage, which were to be composed by a classifier systems architecture. While running controlled by the behavior, a register of the rules being used would be done, containing information from the sensors, the effectors and quality information of the result of the action. This constitute a set of rules mapping sensors to actions and its usefulness, which fits the structure of a classifier, in the classifier systems architecture. Letting the robot run by a certain period of time based only on the behavior, there would be a set of rules memorized, with many of them being valid rules, whilst, eventually, some of them would not be valid. After that time the behavior was to be stopped and the memorized rules to be utilized to start the classifier system architecture. This design would require less effort in developing the behavior and fewer difficulties in evolving the classifiers.

First, the BBR control had to be designed, and the task was rather simple. What happened was that it was possible to design the entire system, leading to safe target seek, without the need of evolving it. It showed that BBR is very well suited for the requirements of target seek considered in this work.

In section 2 it is shown the scope of the work fol-

lowed by a short introductory description of Behavior-Based Robotics in section 3. The robot utilized and its environment is presented in section 4. In section 5, The Architecture Used, the simplicity and elegance of the concepts of BBR is evidenced. In 6 it is shown The Experiment and in 7, Conclusions and Perspectives.

2 The Scope of the Work

The aim of this work is to design a simple architecture based on reactive behaviors for controlling a robot while performing safe exploration of the environment and target seek. The environment is nonlinear focusing real robots in real world. It is developed in simulation to verify the feasibility of the ideas. The simulation, carefully designed to give usable results, is based on a robot configuration and elementary behaviors well tested by Mataric [10].

The BBR approach satisfies the author's requirements of simplicity in design and implementation. It turned out that a basic set of reactive behaviors, developed in accordance with BBR, is very suited to mobile robotics safe environment exploration and, at some extent, as shown in this work, to target seek. This facility gives breathe to think in utilizing evolutionary and / or learning architectures in the development of higher order procedures normally required for a real robot.

Another worth point considered for this work is to show explicitly a solution based on behaviors, because of the so few related material in the literature, in the sense of being help for researchers beginning in the field.

3 Behavior-Based Robotics

To get a comprehensive view and understanding in Behavior-Based Robotics (BBR) it is worth to be in touch with [1]. To get an applied perspective it is worth while to go through [10]; in [12] and [13] it can be found an overview of the field with relevant points related to design and some examples. In "Autonomous Robots" [2] there is a chapter dedicated to architectures, giving a comprehensive summary of the main architectures for autonomous robots including BBR. It's rewarding to read "Flesh and Machines" [6] to get a nice historical vision of artificial intelligence and robotics, written by its principal researcher and leader, who gave the foundations to the BBR. In this section it is presented a very brief introductory note on BBR.

It is required for an autonomous robot to give quick responses and to formulate and follow medium and long run procedures. The state of the art architectures [12], [2] to solve it can take, in a general and simplified overview, three distinct alternatives: deliberative or planner-based, reactive and hybrid. These architectures will be explained in the following sections, followed by BBR, shown somewhere in between of reactive and deliberative architectures.

3.1 Deliberative Architecture

Deliberative or planner-based architectures have a centralized nature and are highly dependent on internal representation. Sensor data are fed and analyzed, in each step, to determine the proper action, being it to deviate immediately from an obstacle or to proceed according to a re-planned path to a long run goal longed for. It gives precise definition for the actions, but the process takes expressive time, and usually will only work in very well controlled environments.

3.2 Reactive Architecture

A Reactive Architecture is characterized by linking tightly sensing to action and, because of that, being able to give quick responses. They achieve real-time performance, in contrast with the Deliberative Architecture's difficulties to accomplish it. But, in the other hand, it only solves immediate purposes of the robot, like object avoidance. It cannot solve medium and long run goals. Its quick response is due mainly to the non use of environmental modeling, but instead, using the environment itself as the model, that is, the robot senses and acts upon the environment directly. This architecture normally doesn't use state and bases its functioning on a mapping between stimuli and appropriate responses. It can also be noticed that the reactive architecture uses little reasoning, whilst deliberative needs a great amount of.

3.3 Hybrid Architecture

First researches in artificial intelligence were mostly deliberative and, more recently, in the last decade of the last century, BBR started solving many of their unsolved problems using reactive behaviors, mainly related to safe locomotion, environment exploration and target seek. One way to take advantage of these two separate and quite opposite techniques is by a hybrid architecture, were a reactive architecture takes care of short run requirements and medium and long run planning are left to a deliberative architecture. This architecture requires usually three layers: Reactive, Deliberative and Intermediate or Supervisory layer, which will take care of conflicts and integration.

3.4 Behavior-Based Robotics

BBR is located somewhere in between Reactive and Deliberative architectures. Besides the basic behaviors being mostly reactive, BBR also can use state and internal representation, being able to deal with immediate, medium and long run goals.

A behavior can be viewed as a procedure or law that performs an action given a condition. For instance, given an object in the near frontal vicinity to the left and none to the right, turn right is an action taken by a behavior of obstacle avoidance.

BBR has a decentralized nature, it is a distributed architecture composed by behaviors that work in parallel, that is, each behavior is fed with sensor data and all the behaviors are processed in each cycle. A behavior can also receive input from others behaviors. Examples of basic behavior for a mobile robot are to avoid obstacles, to follow walls and to seek a target. All of them are fed with sensor data and processed in parallel. Note that they could give conflicting results as the action they suggest, so there must be a conflicting disambiguation mechanism for solving it. This mechanism, depending on the case, can be a hierarchical definition or some combination of the results.

Behaviors in BBR can be designed and implemented incrementally. For example, someone design the behavior 'deviate' to accomplish obstacle deviation. If it's not the first one, he introduces it in the system and implements the procedure to disambiguate possible conflicts with the prior ones. Doing so it's possible to develop and test each behavior to get its right design, independently of finishing the entire project to get it. The new behaviors introduced can take advantage of the previous ones, that is, when it is introduced a behavior for wall following, it stands without dependencies and can take advantage of a previous behavior for obstacle avoidance working in parallel.

Usually a simple and basic behavior is reactive. A great deal of necessities can be solved by reactive behaviors, as can be seen in nature and robotics [4], [5]. But a behavior can have state and environment representation can be implemented by behaviors, in [10] representation is accomplished in a distributed manner, in a network of behaviors. There is a strength performed by BBR researchers to develop more complex behaviors, even social [11], utilizing a combination of base behaviors and learning that gives rise, or emergence, to more sophisticated ones.

By now, what is certain is that BBR is becoming common sense for immediate attitudes of the robot, and even for some medium run tasks like target seek. For more complex problems the way to take is highly dependent on its nature and on the researcher's preferences, there are not yet general rules to decide which way to take, purely BBR or hybrid [1], [2].

4 The Robot and the Environment

In their prior work [15] the authors simulated a robot inspired in the mini robot Khepera [9], world-wide utilized in research. Now the interest is in a robot for living experiences in an office and laboratories. It is a mobile robot with differential wheels, with a circular shape having 60 cm of diameter, with a ring of 12 ultrasonic sensors uniformly distributed. It also has a bearing sensor to detect the direction of the target.

To avoid echoes from the others, only two diametrically opposite ultrasonic sensors are activated simultaneously. So, it's required six activation steps by iteration to get the readings of all the sensors. Before being available to the behaviors modules, the data of the ultrasonic sensors are classified in four ranges: *danger*, *safe*, *edge* and *out*.

The danger range corresponds to proximities that the robot has to avoid; otherwise it can hit the obstacle in the next circle. It's determined by the robot speed and the processing cycle time of the iterations that gives the distance run by the robot until next reading and action. The safe range is used to align to the surface of an object, that's the desired region in which the robot tries to maintain any object. The edge range corresponds to the region the robot avoids objects in its forward direction. Out corresponds to the inexistence of near object.

The bearing measure is provided in each cycle. The robot can be set to accept readings in increments of + - 15 degrees starting from 0, which corresponds to its heading. For instance, if the bearing reading is 67 degrees and the robot is set to accept bearings in the + - 45 degrees range, it'll be rejected, which means that the target is out of view.

Figure 1 shows the robot, a blue circle, surrounded by a drawing of its 12 sonar sensors ranges. The robot is 60 cm of diameter and its heading is shown by a small blue circle. The sonars have a conical aperture of 30 degrees, what gives an almost complete coverage for proximity detection.

The environment is 8 x 10 m, is divided into two rooms and one of them has a table. The target is static and is represented by a small red circle. There is a passage connecting the two rooms. As can be seeing, depending on the position of the robot and the target, the robot has to go trough the passage. Depending on its position relative to the table and the target, the robot will have to escape from the table to get to the target. So, the simulation represents a real situation with real non-linearity given by passage and table.

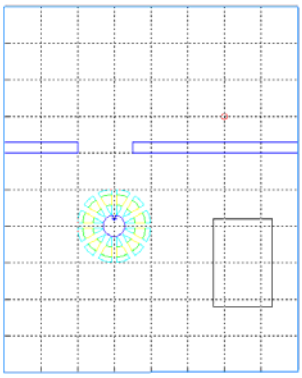

Fig. 1: Robot and Environment
The grid is 1 x 1 m and stands only for measurements.

5 The Architecture Used

5.1 The architecture

The architecture as it is presented here is based on [10]. The behaviors utilized have similarities with the basic ones utilized by Mataric, but are not equal. As shown in [13] the approach to BBR is modular, the behaviors are chosen to be relatively simple and incrementally added to the system, they should not be executed in a serial fashion, but rather in parallel. The design follows a bottom-up procedure resembling biological evolution in its incremental refinements. Behaviors can be activated by external and / or internal conditions, sensory inputs and internal state, but here they will only be activated by external conditions.

More complex behaviors are usually designed indirectly, obtained as an emergent result of the combination of some of the simpler ones. This will be seeing here with environment exploration, obtained from the basic ones: forward, deviate and align. The behaviors constitute a network working in parallel that can be implemented in a distributed manner, through hardware ore software. In software they are easily defined using decision structures like *if <condition> then <action>*, for instance, *if <front left sensor sees an obstacle in a non accepted proximity and the front right sensor is free> then <turn right>*.

The behaviors used here are reactive. The parallel functioning determines that there might be a way to solve conflicts. This will be done letting only one behavior act in each step, by means of hierarchical attributes and exclusive conditions to their activation.

5.2 What the robot has to do

The robot will have to seek a target while performing safe exploration, having a ring of 12 sonar proximity sensors and a way to get the target direction.

For a robot in a unique and empty rectangular room to seek the target would require wall avoidance while seeking, avoiding having priority over seeking. But here there are an obstacle and a passage to another room that puts non-linearity in the solution. Consider the case in Figure 2 where the target is in one room and the robot in the other. With only these two behaviors the robot will reach the wall and will have to deviate from it, and right after will go in the direction of the target again. The combination of these two events will tend to put the robot to oscillate: it is attracted, reaches the wall, it deviates getting off the wall, it is attracted again, and so on. The robot has minimum chance to get to the passage and go through it to the other room. There is nothing in the robot's intelligent control architecture that will put any determination to lead it to the passage, transpose it and reach the target.

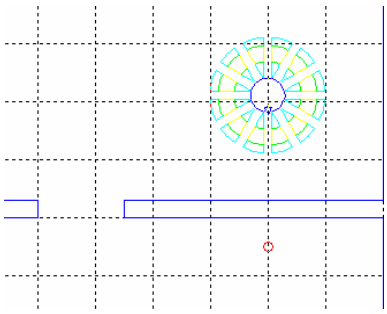

Fig. 2: Non-Linearity

It is required something more then just avoiding obstacles. The robot has to explore safely the environment. How to do that? That's the question that has given a lot of research, many of them requiring hard internal representation and maintenance. It is solved here utilizing only reactive behavior giving the robot the capacity of aligning with the objects it gets near and while it is running. This capacity permits wall following, object aligning and also, that is of great importance, to get through the passage. Now, when the robot, attracted by the target, reaches the vicinity of the wall, it will align with it, having a good chance of getting to the passage and pass through it.

Consider, as the worst case, that once the robot gets close to the wall it will deviate to its left and will have to follow aligned all the inside walls of the room were it is, in a counterclockwise fashion, until it gets to the passage and pass through it. Consider also that while getting through the passage the robot won't be able to sense the target and will take its right, continuing fol-

lowing the walls in a counterclockwise fashion. You can see that it will lead the robot to the target. Actually, it has not to take a so long way, and normally it won't take the shortest way, which, with great effort, would be taken if it was using internal representation. In most cases it will take a good and safe way to the target.

5.3 The Behaviors

The basic behaviors that were developed relate to putting the robot on running if it's safe to do that, to stop it if it is in danger to hit or be hit by any object, to avoid obstacles, to align to objects it gets near and to go towards the target if it is safe to do that. This basic behaviors working in parallel will give the robot the condition to explore the environment safely and get to the target. Environment exploration is guaranteed with high probability due to the fact that aligning with the walls will lead the robot to a passage and it will pass through it. There is a great chance that doing so the robot will safely navigate throughout all the places in the environment. There might be situations in which it gets trapped, as is the case when it gets near an object in the middle of the room and will stay looping around it. But, when it has a purpose, as it happens when it is seeking a target, it will be solved in most of the cases.

Following are shown the behaviors designed to give safe environment exploration. The architecture is presented in the diagram of Figure 3.

Forward: This behavior puts the robot to run if it's safe to do that, or stops it if it is in danger of hitting an object. If any sonar shows an object in the danger region and the front sonars shows an object in the safe and / or danger region, the robot is stopped, will try to find a safe direction to go and, if it fails to find it, it will move the robot backwards in an arbitrary safe direction. The robot will run again when the front sonars find obstacles, if any, only beyond the safe region. The main idea with this behavior is to protect the robot and the environment. It has the highest priority.

Deviate: This behavior is build incrementally over and has low priority then the above one. Case **Forward** is used it will stop and turn aside the robot before putting it to run again. **Deviate** is directed to rapidly avoid an object when the robot is reaching its front edge region, diminishing the need of using **Forward**.

Align: This behavior is built incrementally over and has low priority then the above ones. Its goal is to maintain the proximity, in the safe region, of the object it gets near. If any of the lateral sensors detect an object in the safe range, this behavior is activated and its action will be to bring the robot to the vicinity of the object it is getting near or escaping from.

The behaviors above utilize 30 degrees for each deviation. They give rise to a higher order behavior that emerges from their parallel execution:

Explore: This behavior emerges as the result of the actuation of **Forward**, **Deviate** and **Align.**

To give the robot the ability to seek the target, it was designed the following behavior:

Seek: This behavior is built incrementally over the other ones and its activation is dependent on the safeness of the direction it will have to take to get to the target. The robot senses the bearing of the target and determines the heading to go straight there. **Seek** will check whether it's safe to do it or not, verifying whether the sonars in that direction are free in the danger, safe and edge regions. If it's safe **Seek** will overwrite the heading given by **Explore**. The overture for sensing the incoming bearing can be selected starting in 0 degrees and reaching 360 degrees, in steps of 30 (15 degrees in each side).

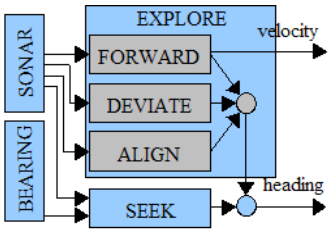

Fig. 3: Architecture

It is important to overwrite the heading of **Explore** because if it's safe the robot doesn't have to follow the contour determined by **Align**, nor the heading giving by **Deviate** or **Forward**. Doing so gives the robot the chance to take safely a straight route to de target. The speed is not overwriting, letting intact the safe decision took by **Forward**.

6 The Experiment

The behaviors were implemented incrementally. These variables had to be adjusted: velocity, bearing sensing aperture, size of the sonars ranges and the time interval of the iterations. The adjustment was empirical and didn't give noticeable difficulties.

As it was said before, the results are surprising. The first important point to notice is the effect of incrementally adding the behaviors, which can be seeing in the trajectories plotted in Figures 4 to 7, where the simulator was put to run 1000 iterations of 1.2 s each with the robot running at 20 cm/s and the bearing sensor with 180 degree aperture. In all of them the robot starts in the middle of the room with the table, looking to the

wall opposite to the passage.

Figure 4 shows the **Forward** behavior working alone. It's easy to identify the sharp changes in orientation that are due to an obstacle getting inside the safe region of the front sonars, in which case the robot is stopped and its orientation is changed until the front sonars are free of obstacles, when it's put to run again. Another important point to consider is that the robot is most of the time located inside one of the rooms, and it can bee seeing only three passage crosses, revealing its difficulty to explore the entire environment.

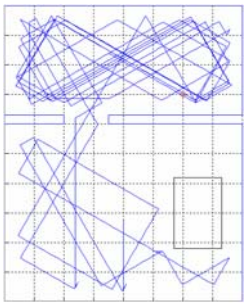

Fig. 4: Forward

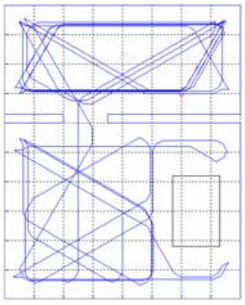

Fig. 5: Forward and Deviate

Figure 5 shows the effect of incrementally adding **Deviate**. It can bee seeing that the changes in orientation are smoother, done in 30 degrees increments. There are still some sharp deviations putting in evidence the importance of **Forward**. There is not yet evidence of exploring efficiently the entire environment.

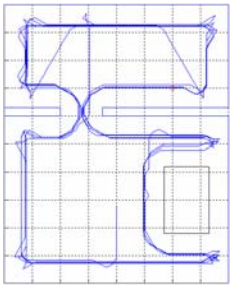

Fig. 6: Forward, Deviate and Align - Safe Exploration

Figure 6 demonstrates the effect of bringing **Align** to work together with **Forward** and **Deviate**, resulting in safe exploration of the environment. It's noticeable the effect in aligning with the objects of **Align**, the smooth changes in orientation of **Deviate** and some stops and sharp orientation changes of **Forward**. The most important is that the robot has a high determination to explore the entire environment, and with safety.

Finally, incrementing the system with target seek the robot achieves the target, as can bee seeing in Figure 7. The robot starts in the middle of a room and the target is in the other. The robot is oriented in an opposite direction related to the target. It will have to re-orient, find a passage, go through it and get to the target, all of these with safety. The robot gets to the target after 108 iterations. In Figure 8, with align turned off, the robot reaches the target after 495 iterations and doesn't show that it has good possibilities to do that.

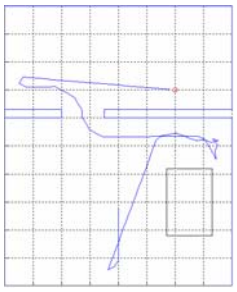

Fig. 7: Target Seek
The target is reached after 108 iterations

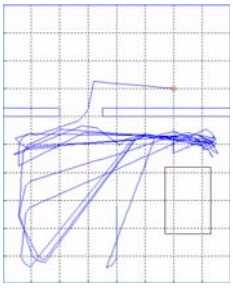

Fig. 8: Target Seek – without Align
The target is reached after 495 iterations.

With align there is not a certainty of reaching the target in a very short run, but the design determines the possibility to do that under a good and safe trajectory, whereas without align the robot is left almost entirely to chance.

7 Conclusions and Perspectives

The results are good considering the purpose of this work, but there is a lot of work to be done to accomplish target seek with minimum efficiency in a real environment. The nature of the design used is not to get the optimum solution, that is, to take the shortest way to the target. But, looking carefully to it, two relevant questions arise: 1) Is it possible to get a shorter way to the target?; 2) Is it possible to guarantee efficiency in a more complex environment, like one having more rooms with alternative paths throughout it?

The authors are working to solve these questions now and will advance some words about. Looking again to Figure 7, it sounds reasonable that if the robot utilized a higher aperture for the bearing sensor it would readily change its direction straight to the target, without having to reach the opposite wall. Again, after crossing the passage, it would be readily attracted to the target. But actually, when it is increased or decreased the aperture, there is the risk of augmenting the chances of oscillation that would result in a longer way or even in diminishing too much the probability of reaching the target. Of course it's possible to optimize the aperture

for a given environment, but that would be very particular and the interest is in a wider solution.

Utilizing landmarks permits to give an affirmative answer not only to the first but also to the second question mentioned above. The authors already developed a theoretical model based on landmarks and pathways using a linear graph, inspired by [10], but with a different basis. Landmarks in strategic positions combined with a simple linear graph of them to define the environment topography can give an efficient way to deal with the non-linearity in a simple or more complex situation. The authors found that the proper positions for landmarks are the passages. To get to the target the robot selects the best linear path linking the landmarks from its actual position to the target. So the robot takes sequentially each of the landmarks as goals to reach in the way to the target. This theoretical model is under implementation through simulation and the authors hope to show its results in the near future.

References:

[1] Arkin, Ronald C. "*Behavior-Based Robotics*", Intelligent Robots and Autonomous Agents Series, The MIT Press, May, 1998.

[2] Bekey, George A. "*Autonomous Robots: From Biological Inspiration to Implementation and Control*", Intelligent Robots and Autonomous Agents Series, The MIT Press, Jun, 2005.

[3] Booker, L. B., Goldberg, D. E. and Holland, John H. "*Classifier Systems and Genetic Algorithms*", Artificial Intelligence, 40 pp 235-282, 1989.

[4] Brooks, R. A. "*Intelligence Without Representation*", Artificial Intelligence Journal (47), pp. 139–159., 1991.

[5] Brooks, R. A. "*Intelligence Without Reason*", Proc 12th Int. Joint Conf on A Intelligence, August, 1991.

[6] Brooks, Rodney A. "*Flesh and Machines: How Robots Will Change Us*", Pantheon Books, Feb, 2002.

[7] Cazangi, Renato R. "*Uma Proposta Evolutiva para Controle Inteligente em Navegação Autônoma de Robôs*", Master of Science Dissertation in Electrical Engineering, UNICAMP, May, 2004.

[8] Holland, John H. "Adaptation *in Natural and Artificial Systems*", University of Michigan Press, 1975

[9] Khepera, produced by K-TEAM, http://www.cyberbotics.com/products/robots/khepera.html".

[10] Mataric, Maja J. "*A Distributed Model for Mobile Robot Environment-Learning and Navigation*", Master of Science Thesis in Electrical Engineering and Computer Science, Technical Report AI-TR-1228, MIT Artificial Intelligence Laboratory, May, 1990.

[11] Mataric, Maja J. "*Interaction and Intelligent Behavior*", Doctorate Thesis in Electrical Engineering and Computer Science, Technical Report AI-TR-1495, MIT Artificial Intelligence Laboratory, May, 1994.

[12] Mataric, Maja J. "*Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior*", Journal of Exp. and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents, 9(2-3), pp. 323-336, H. Hexmoor, I. Horswill, and D. Kortenkamp, eds., 1997.

[13] Mataric, Maja J. "*Behavior-Based Robotics as a Tool for Synthesis of Artificial* Behavior *and Analysis of Natural Behavior*", Trends in Cognitive Science, pp. 82-87, Mar, 1998.

[14] Mataric, Maja J. and Cliff, Dave "*Challenges In Evolving Controllers for Physical Robots*", in "Evolutional Robotics", special issue of Robotics and Autonomous Systems, 19(1), pp. 67-83, Oct, 1996.

[15] Moussi, Lubnen N. "*Aplicações de Sistemas Classificadores para Robótica Autônoma Móvel com Aprendizado*", Master of Science Dissertation in Electrical Engineering, UNICAMP, Nov, 2002.

[16] Moussi, Lubnen N., Gudwin, R. R., Von Zuben, F. J. and Madrid, M. K. "*Neural networks in classifier systems (NNCS): An application to autonomous navigation*", in V. V. Kluev & N. E. Mastorakis (eds.) Advances in Signal Processing, Robotics and Communications, Electrical and Computer Engineering Series, pp. 256-262, WSES Press, 2001.

[17] Moussi, Lubnen N., Gudwin, R. R., Von Zuben, F. J. and Madrid, M. K. "*A Simulator using Classifier Systems with Neural Networks for Autonomous Robot Navigation*", proc IEEE International Joint Conference on Neural Networks (IJCNN'2002), vol. 1, pp. 501-506, in 2002 IEEE World Congress on Comp. Intelligence (WCCI'2002), May, 2002.

[18] Nolfi, S. "*Evorobot 1.1*", a software for running evolutionary robotics that can be downloaded free from. http://gral.ip.rm.cnr.it/evorobot/simulator.html.

[19] Nolfi, S. and Floreano, D. "*Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*", Intelligent Robots and Autonomous Agents Series, The MIT Press, Nov, 2000.