# Automatic Vehicle Parking Using an Evolution-Obtained Neural Controller

Franco Ronchetti, Laura Lanzarini

Institute of Research in Computer Science III-LIDI (III-LIDI)
National University of La Plata
La Plata, Buenos Aires, Argentina
{fronchetti,laural}@lidi.info.unlp.edu.ar

**Abstract.** Within the problems that can be solved with autonomous robots, automatic parking is an area of great interest, since it presents a complex scenario where the agent must go through a series of obstacles to reach its goal. Existing solutions usually require some kind of external mark for monitoring or global vision that indicates where the agent is at a given time. This article presents an evolutionary strategy to generate a robotic controller based on a neural network that successfully solves the problem of vehicle parallel parking using only local information. The performance of the fitness function is analyzed, focusing not only on the agent reaching its goal, but also on it doing so in a manner that is appropriate for the physics of a vehicle. Additionally, the Player/Stage simulator is broadly discussed, since it is one of the most widely used simulators nowadays in robotics.

**Keywords:** Evolutionary robotics, neuroevolution, automatic parking

## 1 Introduction and Related Works

Automotive companies have shown a strong interest in incorporating automatic parking to the set of features offered in their vehicles. This problem, which belongs to the area of robotics, already has some partial solutions available in the market. There are high-end cars that have intelligent technology that controls the movements of the steering wheel during parking. However, in this case in particular, the driver is still responsible for the maneuver, operating the accelerator and the brakes [1].

If the existing literature is analyzed, it can be observed that the central aspect that conditions how the controller is obtained is related to the information available to determine the position of the vehicle. There are basically two alternatives: global vision or local sensors placed on the periphery of the vehicle. In general, the first alternative is available in more expensive vehicles, while local sensors can be found in medium-cost segments and above.

The strategies proposed are varied. Among the solutions that use the global vision, [2] and [3] can be mentioned, where the automatic parking problem is solved as if it were a geometry and kinematics problem rather than a control one, or [4], that defines a controller by combining a neural network with diffuse rules. In [5], local information is used (such as an image from a CCD) to look for marks or tracks on the floor. This simplifies the resolution of the problem by limiting its application to the resolution of real problems. Some works, such as [6] or [7], use diffuse logics systems, which in general requires additional work to obtain the diffuse sets beforehand.

Neural networks have proven to be a highly useful tool to define robotic controllers because they have the ability of representing the knowledge acquired through a structure that, once trained, can operate in real time [8][9][10][11].

As regards how to obtain a neural controller with the ability of parking a vehicle, there are solutions based on neural networks that carry out the adaptation by means of a training algorithm [12] [4]. These processes require a set of patterns, generated by some expert, accurate enough for the robot to be able to perform its task and broad enough to achieve a good generalization of the controller. It should be mentioned that these solutions focus only on getting the controller reach its goal, but they do not take vehicle physical aspects into consideration.

In this paper, a solution is proposed that is based on a neural controller whose adaptation is done through evolution rather than training. The method uses local information only, and it does not need previous information to guide controller behavior. The adaptive process is of interest not only in that the controller is able to solve the problem, but also that it does so in an appropriate manner.

## 2 The Simulator

The adaptation of a robotic controller through an evolutionary strategy is computationally costly if done in a real robot. For this reason, a simulator that reduces the time required to measure the fitness of the various controllers that are generated during this process is required.

For this work, a scenario was configured on the Player/Stage platform because this is the Open Source simulator most widely used in the field of robotics [13]. The platform correctly simulates, in real or accelerated time, a kinematics system with various types of sensors and triggers, such as proximity IR sensors. In addition to the various, already existing models, the simulator allows designing a customized two-dimensional robot with its sensors. A test scenario where different walls or obstacles are defined can also be designed. Figure 1-a shows the model used positioned on a fragment of the scenario created for this problem.

It should be noted that even if the perception of reality by the robot increases with the number of sensors, the complexity of the neural controller also increases. Therefore, various experiments were carried out, and it was decided to leave the lowest possible number of sensors required to solve the problem. Figure 1-a shows the 10 sensors used numbered from 0 to 9.

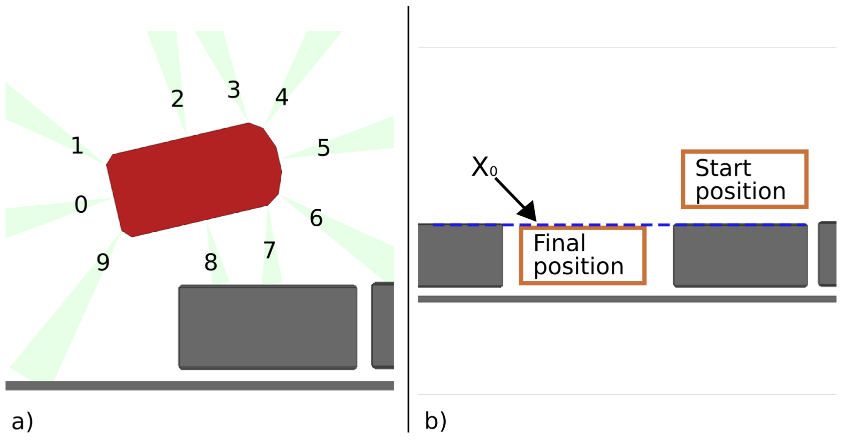

**Fig. 1.** Model and scenario simulated in Stage. a) Vehicle with its 10 IR sensors. b) Diagram of the initial and final positions of the route.

## 3 Description of the Problem

Automatic parking belongs to a category of robotic problems called “homing”. Here, the problem consists in a robot that has to reach a certain objective (or “homing”) from a fixed position, or from any location on a given scenario. Vehicles use a kinematic system known as *Ackermann*, which uses two back wheels for traction and two front wheels to establish the turning angle. This system is different from the traditional *differential* system used in most of scientific-use robots.

In this work, the vehicle started at an initial position known as *ready-to-reverse*. To define the target position, we first need to define the meaning of ”parking a vehicle”. In this context, the final position was not defined as an absolute coordinate, but as an equation of the straight line in the system $(x, y, \theta)$, where $x, y$ are the coordinates of the robot and $\theta$ is the inclination angle. A vehicle is considered to have been parked correctly when its position is equal to the variables (with an error margin) $(x_0, y, \theta_0)$. That is, the agent must approximate variables $x$ and $\theta$, freeing variable $y$. Figure 1-b shows this event.

## 4 Strategy Proposed

The strategy developed is a populational metaheuristic based on an evolutionary algorithm equivalent to the one proposed in [8]. The controller that is generated is composed only by a neural network that controls the movements to be made. The network receives local information and produces two outputs: one indicating the speed of the vehicle, and another one indicating the turning angle for the wheels. Each individual in the population has, in addition to other parameters required for the process, a fixed-architecture neural network. That is, the evolutionary process focuses on the modification of weights on the network. The process uses

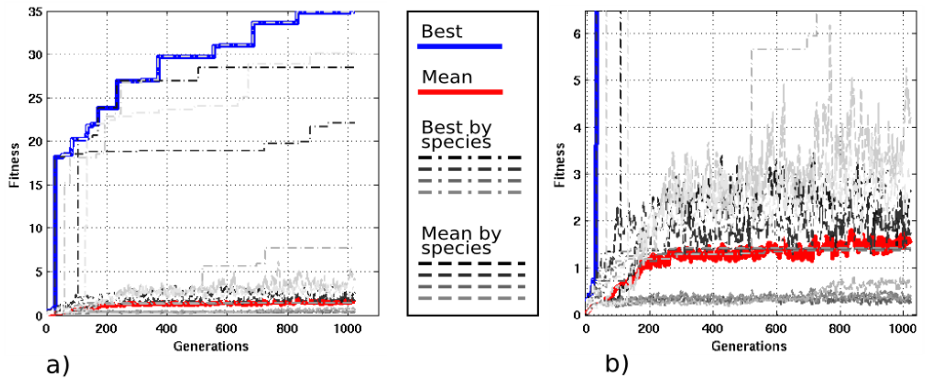

**Fig. 2.** Fitness behavior of the various species in an execution. a) Original chart. b) Enlargement reducing fitness scale.

speciation as tool to avoid local optima. Here, different groups of individuals are grouped when the program starts based on their chromosome. Then, each species is developed independently from the others, with an intervention among them every certain number of generations. Various works have shown that the exploration of the search space in a semi-parallel fashion markedly increases the algorithm execution time and solves the search for the global optimum more efficiently. An example can be found in [10]. The time during which species remain isolated must be balanced. If this time is too long, a merely parallel search may result. On the other hand, if it is too short, it would be the same as having only one species. Figure 2 shows the behavior of the various species throughout a specific execution. It can be seen how each species reaches different optima.

To generate a new population of individuals, a probabilistic binary tournament is used, where a selection of candidates is created. Then, mutation parameters are applied so that the new generation is a group of clones of the candidates but with small modifications. In order to retain the fittest individuals, an 8% elitism per species is used.

This evolutionary strategy does not replace a generation with the next one, as genetic algorithms usually do, but it rather uses an algorithm with variable-size population. To carry this out, individual life times must be properly assigned. In this case, the method proposed in [14] was used.

## 4.1 Neural Architecture

Neuronal networks are computational models with a great generalization capacity, highly efficient for the resolution of two types of problems: clustering and mapping functions which, given a certain input, produce the expected output. For this reason, the most famous architectures do not usually take into account temporal factors. By using a neural controller to command an autonomous robot, some kind of memory must be implemented, since one data set must be interpreted in different ways at different times. To achieve this, recurrences are usually introduced in network connections. That is, there can be connections from one network layer to a previous one, lateral connections, or even connections within one neuron.

There are few conclusive studies that define which architectures should be used for certain problems where time is an essential variable. In [10], a hidden layer of the network dedicated to controller memory is proposed. In [11], fully connected architectures are used on the grounds that a "smaller" architecture can be obtained by setting to zero certain arches of the network.

For this paper, two parameters that provide temporal information to the network were used: network outputs of $t$ units of time backwards, and an *odometer*, which provides the network with information about the distance traveled by the robot at a given time. The latter is an escalated value to prevent the network from being flooded by entering with high values. The rest of the network works as a conventional feed-forward network.

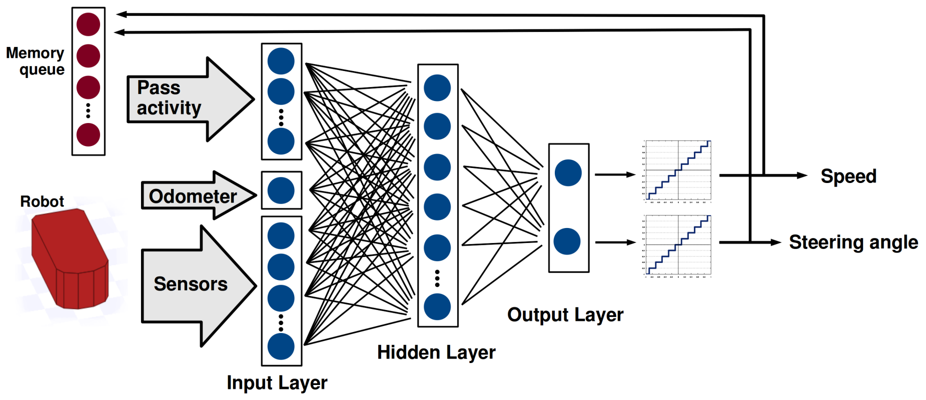

**Fig. 3.** Generic network architecture of the neural network used.

All neurons use *hyperbolic tangent* transfer functions, which allows limiting network output values. Output neurons use a composite transfer function, where after applying the *hyperbolic tangent* function, the following tiered function is applied:

$$netOut = round(netOut * 10)/10; \qquad (1)$$

Thus, the network output is transformed to some 21 discrete values (10 positive, 10 negative, and zero). This helps different situations when driving the vehicle.

To set the speed and the turning angle of the agent, network outputs are multiplied by the values 3 and 45, respectively. The tiered function defined in (1), together with these limits, allows having an approximate resolution of $0.25km/h$ for speed and 4 for turning angle. Figure 3 summarizes the general structure of the neural network used.

For this work, the following architecture was used:

– **17 input neurons.** 10 sensors, 1 odometer, and 6 neurons that indicate the recent past (3 for output unit 1, and 3 for output unit 2).
– **10 hidden neurons.**
– **2 output neurons.** One to control speed and one to control turning angle.

## 4.2 Fitness function

Generally, when using a population evolutionary process, the metaheuristic controlling population movements in the solution space receives special attention, but the fitness function used is not particularly considered. [15] includes a survey and analysis of various fitness functions used in works related to evolutionary robotics.

To carry out a fitness function that evaluates the good performance of an autonomous robot when maneuvering to park, the following aspects have to be considered:

1. The agent reaching its objective.
2. Smooth movements. It is undesirable that a vehicle changes its speed or turning angle abruptly.
3. Parking with the least possible number of maneuvers. Each time the agent changes the sign of its speed is a *maneuver.*
4. Travelling the least possible distance.

The first aspect is the main objective, so the individuals that do approach the destination must be rewarded. The analysis of aspect #2 is important, since in general, predicting the behavior of a neural network in time is difficult. The third aspect, the same as the second, is related to the correct behavior of a vehicle. The last one is an additional improvement but important nonetheless, since it would not be desirable that a vehicle travels too many meters before maneuvering.

**Description of the Function** The function developed for this paper, the same as all functions used in evolutionary robotics, evaluates the behavior of the robot in a series of steps. Here, the number of steps is given by time. The robot is evaluated a maximum of 60 simulated seconds, or until it collides against an obstacle.

When the evaluation loop ends, the fitness value is determined as the average behavior during the evaluation, where this behavior is defined by the robot performing smooth movements. Then, fitness is increased inversely proportionally to the distance to the objective. Additionally, fitness is considerably reduced if the robot collides. The function has 3 evolutionary stages that orient individuals:

- During the first stage, individuals are rewarded for approaching the objective. This is done when the evaluation process ends.
- After an individual reaches its target, the desired behavior is for it to stop. For this reason, individuals are rewarded as they reduce their speed.
- Finally, when an individual effectively parks, the best solution is filtered based on the number of maneuvers done and the distance traveled.

```
$fitness = 0; numSteps = 0; numChanges = 0; targetReached = false;$
while (simulationTime $\leq 60$) AND (notCrash) do
    $numSteps = numSteps + 1$
    Read sensor data, compute ANN, and update memory
    if Changing the direction of movement then
        $numChanges = numChanges + 1$
    end if
    $softMov$ = Difference between last and current outputs
    $fitness = fitness + 0.4^{softMov}$
    if $targetReached$ then
        $fitness = fitness + 0.1^{|robotSpeed|}$
    else
        if $distanceToTarget \leq validError$ then
            $targetReached = true$
            10 more seconds to evaluate
        end if
    end if
end while
$fitness = fitness/numSteps;$
$fitness = fitness * 0.5^{distanceToTarget-1}$
if $crash$ then
    $fitness = fitness * 0.3;$
else
    if ($distanceToTarget \leq validError$) AND ($robotSpeed = 0$) then
        $fitness = fitness * (0.8^{trajectoryDistance-8} + 1) * (0.8^{numChanges-2} + 1)$
    end if
end if
```

**Fig. 4.** Pseudo-code of the fitness function.

Figure 4 presents the pseudo-code of the function described here. All exponential expressions grant a higher fitness the smaller the input data. Each expression is customized for the corresponding values. The function is called 3 times by each individual, positioning the robot in different places and different parking spaces. This allows the neural network to perform a correct generalization. It is considered that a vehicle reaches its target when the distance to it is less than 0.2 meters, and it is considered that a vehicle parks when it reaches its target and its speed is null.

## 5 Experimental Results

First, a communications library was implemented to be able to use *Matlab* with the Player/Stage platform. The evolutionary process is fully programmed in this high-level language and with the help of the neural network toolbox it provides.

Various experiments were carried out with the implemented neural architecture. A scenario with various spaces and parking manners was implemented for network behavior generalization. Similarly, various tests were carried out considering different vehicle proximity sensor number, position and angle.

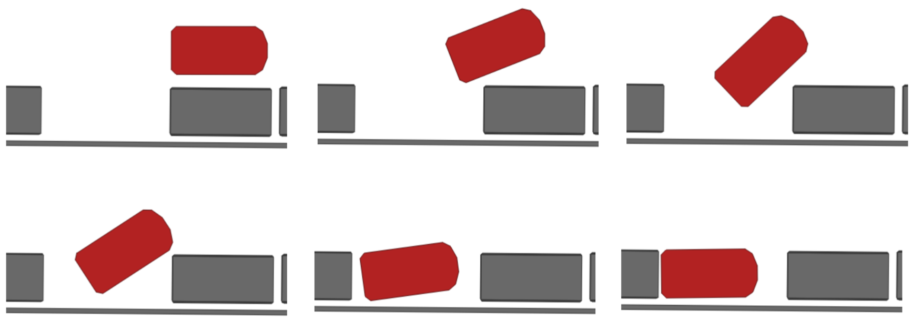

**Fig. 5.** Maneuver carried out by the best individual in the evolutionary process, in the smallest parking space.

By optimizing the weights of a neural network, the cardinality of the solution space is very high, so a great genetic diversity is required. For all experiments, an initial population of 400 individuals created with a normal distribution with mean 0 and standard deviation 0.5 grouped in 10 species was used. The mutation parameters used ranged between 0.2 and 0.3, with a mutation degree between 0.3 and 0.03 [8]. The best controllers obtained in 50 separate runs of the evolutionary process were analyzed, and it was verified that 100% of them parked correctly with an error of $\pm 25cm$.

Figure 5 shows the execution of the best controller obtained after 900 generations in a reduced parking space. Figure 6, on the other hand, shows the behavior of the agent during the maneuver. The chart to the left (a) shows network output in time, represented by the number of evaluations (approximately 4 per second). The controller behaves as desired, with smooth movement changes. The chart on the right (b) shows the distance between the vehicle and the desired destination. It should be noted that when the vehicle finishes maneuvering, speed becomes null.

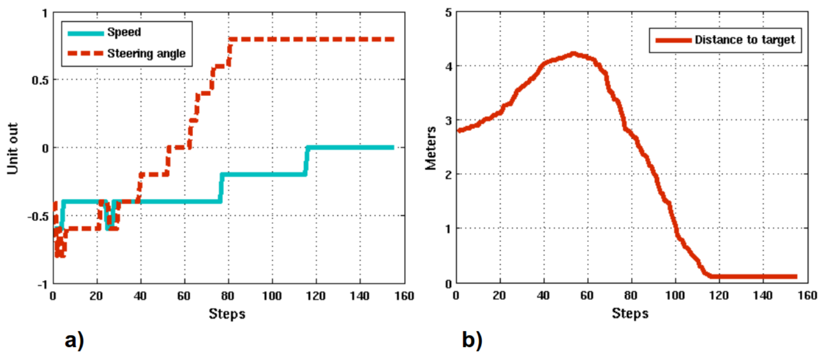

**Fig. 6.** Performance of the robot when maneuvering for parking. a) Neural network outputs. b) Distance to the objective.

## 6 Conclusions and Future Work

A method was developed for obtaining a neural robotic controller that allows maneuvering for parallel parking. The study of an appropriate neural architecture and the handling of temporal information are of vital importance in evolutionary robotics problems.

The controller obtained was able to successfully generalize the maneuver for different parking spaces, using only local information from the vehicle. This allows taking the controller to the practice with a real agent.

The versatility of the Player/Stage platform allows carrying out optimization studies at the positions of the proximity sensors, based on conditions of real vehicles.

Currently, work is being done on the strategy to solve various additional automatic parking problems, such as:

- Use of the controller developed as module to generate a behavior that is more robust and allows different parking maneuvers, such as 90 parking, or searching for the *ready-to-reverse* position.
- Parking in reduced spaces that require many maneuvers. In general, there are only a few works that were able to support this problem, since a detailed analysis of different situations is required.
- Decision on the best maneuver to use, so that the agent can select different controllers to perform the best maneuver for the problem at hand.

## References

1. Ford Technology. Active park assist. hassle-free parallel parking. `http://www.ford.com/technology/`.

2. D. Maravall and J. de Lope. Multi-objective dynamic optimization with genetic algorithms for automatic parking. *Soft Comput. A Fusion of Foundations, Methodologies and Applications.*, 11:249–257, 2006.
3. J-Y. Lee, M-S. Kim, and J-J. Lee. Design of fuzzy controller for car parking problem using evolutionary multi-objective optimization approach. *IEEE International Symposium on Industrial Electronics.*, 2006.
4. W.A. Daxwanger and G.K. Schmidt. Skill-based visual parking control using neural and fuzzy networks. *IEEE International Conference on Systems, Man and Cybernetics.*, 2:1659–1664, 1995.
5. J. Xu, G. Chen, and M. Xie. Vision-guided automatic parking for smart car. *IEEE Intelligent Vehicles Symposium.*, 2000.
6. Y. Zhao and E. G. Collins Jr. Robust automatic parallel parking in tight spaces via fuzzy logic. *Robotics and Autonomous Systems*, 51(2-3):111–127, 2005.
7. Y-W. Ryu, S-Y. Oh, and S-Y. Kim. Robust automatic parking without odometry using an evolutionary fuzzy logic controller. *International Journal of Control Automation and System*, 6(3), 2008.
8. F. Ronchetti and L. Lanzarini. Robotic controller obtained through a speciation-based metaheuristic. *INNS International Educational Symposium on Neural Networks*, 2011.
9. L. Lanzarini, G. Osella Massa, and H. Vinuesa. Modular creation of neuronal networks for autonomous robot control. *Revista Iberoamericana de Inteligencia Artificial*, 11(35):43–53, 2007.
10. G. Capi and K. Doya. Evolution of recurrent neural controllers using an extended parallel genetic algorithm. *Robotics and Autonomous Systems*, 52:148–159, 2005.
11. A. L. Nelson, E. Grant, J.M. Galeotti, and S. Rhody. Maze exploration behaviors using an integrated evolutionary robotics environment. In *Journal of Robotics and Autonomous Systems*, pages 159–173, 2004.
12. H. Milton, F.S. Osrio, F. Heinen, and C. Kelber. Seva3d: Using artificial neural networks to autonomous vehicle parking control. *IEEE WCCI (World Conference on Computational Intelligence)*, 2006.
13. R. Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2-4):189–208, 2008.
14. L. Lanzarini, C. Sanz, M.Naiouf, and F. Romero. Mixed alternative in the assignment by classes vs conventional methods for calculation of individuals lifetime in gavaps. *Proceedings of the 22nd International Conference on Information Technology Interfaces, ITI 2000*, 953-96769-1-6:383–389, 2000.
15. A. L. Nelson, G. J. Barlow, and L. Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370, 2009.