Autonomous Vehicle Parking and Pull Out using Artificial Neural Networks

Milton Roberto Heinen, Fernando Santos Osório, Farlei josé Heinen and Christian Kelber
Universidade do Vale do Rio dos Sinos (UNISINOS) - Applied Computing
mheinen@turing.unisinos.br, {fosorio,farlei}@unisinos.br, kelber@eletrica.unisinos.br

Abstract

This paper describes a simulation system proposed to accomplish in a realistic way an autonomous parking in a parallel parking space, and also to pull out the vehicle of the parking space to go back to the traffic lane. The proposed system controls a car-like vehicle based on the sonar sensors readings, and using an Jordan neural network to automatically generate acceleration and steering commands. The results obtained in our simulations demonstrated that the proposed controller is perfectly able to correctly park and pull out the vehicle.

1 Introduction

The autonomous vehicles and robots (Autonomous Mobile Robots - AMR) have attracted the attention of a great number of researchers, mainly due to the great challenge that this new domain of research offers: to endow these systems with an intelligent reasoning capability, exploring their abilities to interact with the environment where they are inserted[1]. AMRs should perceive the environment through their sensors (e.g. infrared, sonar, lasers, video cameras), and from them they can be able to plan and execute their actions[7].

Starting from the studies and research work developed by the Artificial Intelligence Group (GIA-PIPCA) and by Autonomous Vehicles Group (GPVA) at Unisinos[1] related to the development of applications in the area of mobile autonomous robots, the basis of this work were created. We should notice particularly the initial development of the SEVA3D[2] system (Autonomous Vehicles Parking Simulator in a three-dimensional environment)[10, 13]. Our main goal was to develop a system capable of accomplishing virtual realistic 3D simulations, implementing a better simulation tool of autonomous parking in parallel parking spaces. This system uses a realistic three-dimensional environment model with a 3D sonar sensor model, which includes noise as in real sonar data. Therefore SEVA3D is much closer to the reality, and the simulation results will be easily transposed to our automated Mini-Baja Buggy developed by the GPVA Group at Unisinos[14].

The control system implemented in SEVA3D-A accomplishes the parking of vehicles, controlling them in an autonomous way, using a finite state automaton (FSA). The implemented system is also capable of pull out the vehicle of the parking space to go back to the traffic lane. The experiments using SEVA3D-A worked in a satisfactory way, but we needed to code manually all the rules used to define the FSA functioning. The task of specifying FSA rules has proven to be a difficult task, that needs a lot of a priori knowledge about the system functioning, and resulting a few robust vehicle control behavior in non expected situations.

For these reasons, we decided to develop a new version of the system SEVA3D, called SEVA3D-N[12, 11]. The vehicle parking and pull out control in SEVA3D-N is accomplished using a neural network inspired in the Jordan-net model. The main advantage of this approach is the automatic knowledge acquisition instead of code it manually. The neural net is able to learn how to control the vehicle from examples, so we just need to show how to park and pull out the vehicle - showing how to do it, instead of specifying how to control it. In our previous work[12], we autonomously park the vehicle in the parking space. In this paper, we describe a new version of SEVA3D-N, that is capable to pull out the vehicle back to the road lane.

2 Related works

Studies related to driving assistance systems and autonomous parking of vehicles have been done by some research groups and companies. Among them, one of

[1] http://www.exatec.unisinos.br/~autonom/
[2] http://inf.unisinos.br/~osorio/seva3d/

the first outstanding studies were accomplished by INRIA researchers[5, 16, 18], which implemented a control system used to park an adapted Ligier electric vehicle in an autonomous way. This vehicle was equipped with 14 sonar sensors and it was used a manually customized set of rules to accomplish the task. In order to become possible to measure in advance the parking space depth, it was necessary to install a barrier of moderate height close to the curb. The introduction of this artificial barrier turns possible to determine more precisely the parking space depth using the available sensors[18].

The drawbacks of this approach are: (i) the curb barrier installation requirement, which restricts the practical application of this method on conventional streets; (ii) the great number of sonar sensors that must be installed around the vehicle (fourteen sensors); (iii) the limited application of this algorithm that works specifically in parallel parking tasks and needs to be manually coded. We aim to propose a new system that uses a small number of sensors, should be able to be used in conventional parking spaces, and also must allow the adaptation to different parking situations. The SEVA3D-N is the system we developed and we are improving in order to achieve these goals.

3 SEVA3D simulator

The SEVA3D-N Simulator possesses several improvements related to the SEVA3D-A. The major advantage of SEVA3D-N is the fact that using an artificial neural network we do not need to manually code vehicle control rules (FSA), once the control system is obtained by training a neural net. We just need to show some examples of parking and pull out maneuvers execution and the system will automatically adapt the network parameters, learning to autonomously park and pull out the vehicle. The main components of the SEVA3D simulator are:

- Perception Module: measures the distance from obstacles based on a sonar sensor simulation;
- Action Module: sends action commands to vehicle actuators, defines the direction (forward, backward), the speed (gas pedal control) and the car orientation (steering wheel rotation);
- Kinematics Module: uses the Ackerman model to estimate the vehicle trajectory, considering vehicle direction, speed and steering wheel angle;
- Control Module: receives sensor data from the Perception Module and sends actions to the Vehicle Action Module, using a neural network to control the parking and pull out maneuver.

3.1 SimRob3D simulation tool

The implementation of SEVA3D uses the SimRob3D simulation tools[8, 9], previously developed by our group. The SimRob3D tools main characteristics are: to provide 3D environment visualization tools used in simulations, and to provide customizable mobile robots simulation tools. The 3D environment objects can be modeled using different 3D modeling software, and allows to detail the different elements present in the environment (vehicles, streets and buildings), resulting in a high level of realism. The SimRob3D tools also include different sensorial and kinematics models, which can be used to customize the simulated mobile robots.

3.2 Perception module

Sonars are a common distance sensor used in robotic applications, because they can estimate with a reasonable precision the distance from objects positioned near to them[1]. Sonars can be used to perceive the environment objects and obstacles, and they offer a good precision/price rate related to other distance measuring methods[3].

The simulated sonar sensors allow to estimate the distance between the vehicle and the obstacles present in the environment: other cars and the edge of the sidewalks. The six sensors used were distributed in strategic positions of the vehicle, as showed in Figure 1. In our

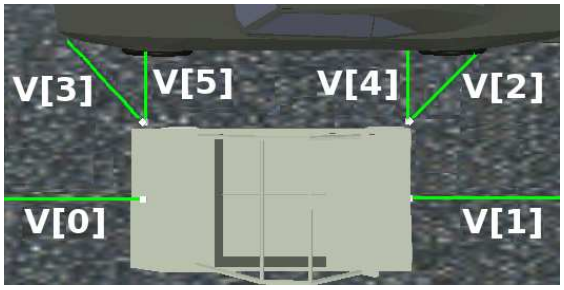


Figure 1. Distribution of the sonar sensors

experiments we put sensors only in one side of the vehicle, because all experiments were developed in order to park the vehicle in parallel parking spaces located on the right side of the car, which is a typical situation in two-way streets.

The SimRob3D sonars are simulated through the definition of a conical section of the virtual space, where the objects which remain inside this volume can be detected. The intersection between objects and the sonar cone volume (perceptual space) is detected using a stochastic approach. Several object detection lines (rays) are gener-

ated from the position of the sensor directed according to the sonar spatial orientation, remaining inside the sonar cone volume. A RayCast[3] technique was used to generate rays, that are randomly distributed in the sonar cone volume. If any of them collide (intersect) with some object polygon, the distance from the sensor until the collision point is informed.

Besides the sonar sensors, sometimes it was also necessary to use an odometer. The odometer was used only to verify if the parking space is enough to allow the correct parking of the vehicle, when there are no other parked cars available to be used as reference points. This situation occurs when there are large empty parking spaces.

3.3 Kinematics Module

The movement of the vehicle respects the Ackerman kinematics model[3], which was also the model adopted in the precursor work developed at INRIA[5]. In this model a simulated vehicle is represented by a rectangular volume supported by four wheels divided in two axes, where the back wheels are attached to a fixed axis and the front wheels can be turned, controlling them through the steering wheel[1]. The Figure 2 shows the elements of the kinematics model.

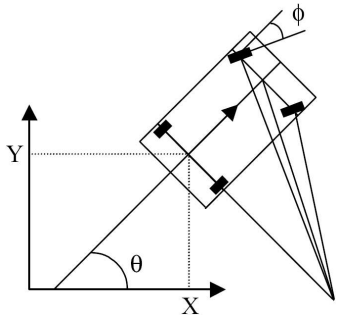


Figure 2. Kinematics model

The vehicle's location (position and orientation) relative to some reference coordinate system is denoted as $q = (x, y, \theta)^T$ where $x = x(t)$ and $y = y(t)$ are the coordinates of the rear axle midpoint, $\theta = \theta(t)$ is the orientation of the vehicle, and t is time. The motion of the vehicle is described by the following equations[16]:

$$\begin{cases} \dot{x} = v\,\cos\phi\,\cos\theta, \\ \dot{y} = v\,\cos\phi\,\sin\theta, \\ \dot{\theta} = \frac{v}{L}\,\sin\phi, \end{cases} \quad (1)$$

where $\phi = \phi(t)$ is the steering angle, $v = v(t)$ is the locomotion velocity of the midpoint of the front wheel axle, and L is the wheel base. The steering angle and locomotion velocity are two control commands (ϕ, v). Eqs. 1 correspond to a system with non-holonomic constraints because they involve the derivatives of the coordinates of the vehicle and are non-integrable[15].

[3]RayCast is a computer graphic technique that simulates the physical effects associated with the propagation of light rays[4]

3.4 FSA control - parking

In SEVA3D-A, the vehicle control task is accomplished by a Finite State Automaton (FSA). The Figure 3 shows the diagram of the finite state machine used to control the vehicle, and therefore this successful FSA was also used to be learned by an artificial neural network in SEVA3D-N. The following states were defined:

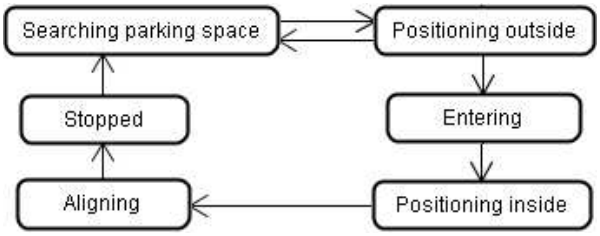


Figure 3. FSA control - parking

Stopped: automaton initial and final state;
Searching for parking space: the vehicle moves in a straight forward direction, searching for a free parking space. If an available parking space was found then the FSA state changes to *Positioning outside*;
Positioning outside: the vehicle moves forward in order to reach a correct position to start entering in the parking space. This state is also used to verify if the parking space is adequate. If the space is too small, the state returns to *Searching for parking space*. If the space is large enough, the state changes to *Entering*;
Entering: the car starts to move backward and the steering wheel is turned to the right, entering in the parking space. When the sensor **V[2]** detects the sidewalk curb the state changes to *Positioning inside*;
Positioning inside: the vehicle continues to move backward, but the steering wheel is turned to the left. When the sensor **V[3]** detects the sidewalk curb the state changes to *Aligning*;
Aligning: the vehicle is moved in order to reach an adequate distance from the cars parked ahead and behind it. After the alignment, the state changes to *Stopped* and the maneuver is terminated.

3.5 FSA control - pull out

The Figure 4 shows the diagram of the finite state machine used to pull out the vehicle of the parking space. The following states were defined:

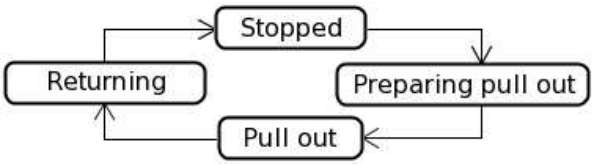


Figure 4. FSA control - pull out

Preparing pull out: the car starts to move in a straight backward direction, until the **V[1]** detects a near obstacle (another parked car) or until the frontal space is sufficient to going out;
Pull out: the vehicle moves in a forward direction executing a "s" shaped trajectory, until the sensors indicate the car is outside of the parking space;
Returning: the car is aligned in a parallel way to the lane, the state changes to *Stopped* and the maneuver is terminated.

3.6 Neural control

The results we obtained with the control based on a manually specified FSA, adopted in SEVA3D-A, motivated us to study other alternatives to the parking control task knowledge acquisition. The FSA creation process was difficult, and rule codification was an arduous task and besides that the final result does not guarantee a great robustness to the system. As the control system was hard coded, the FSA needs a fine tuning every time we have some change in sensors and actuators behavior and also if an unexpected situation occurs (e.g. input noise, imprecise actuators). We looked for a practical solution that should be capable to automatically learn how to control the vehicle. The adopted solution was the use of artificial neural nets (ANNs)[20] with supervised learning, which are capable to adapt themselves to different situations. The ANNs can learn how to control the vehicle in a parallel parking task from a set of practical examples: we just need to show how to park and pull out the vehicle.

The first step was to create the parking examples data set to be employed in the ANN learning. The SEVA3D-A simulator was adapted in order to generate a log file, containing records of: the sensors state, the FSA state and the commands sent to the vehicle actuators (speed and steering wheel angle). The generated file allowed to train a neural net, and we expected that the resulting ANN could be able to reproduce the FSA behavior. This initial experiment worked fine and later SEVA3D was also used to generate a new set of parking task examples, with the vehicle being controlled by a human instead using the autonomous control system.

The ANN adopted model was a Jordan-net[17], a modified version of Multi-Layer Perceptron (MLP) nets[6] with recurrent inputs. The learning algorithm used was the Resilient Propagation[19]. This network was used in the following way: a set of inputs are used to indicate the current state of the network (representing the active FSA state) and another set of outputs are used to indicate the next state of the network (the FSA state in the next cycle). The next state indicated by the network outputs can be the same as the current state or it can be changed in order to assume a new network state. The network state changes occur in function of their inputs represented by the sensors plus the current state. So, the neural net receives as inputs the sensors data and the current state (Figure 5) and based on this information decides if the current state remains the same, or if it is time to change to a new state.

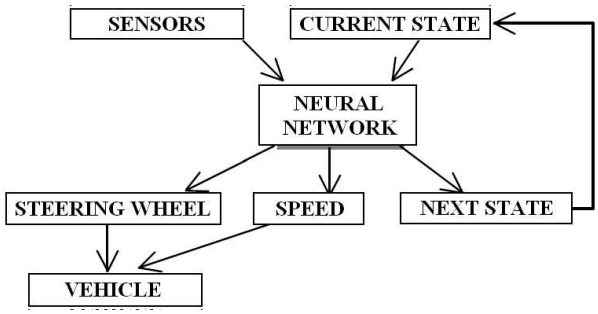


Figure 5. Neural network model scheme

The next state (current or new), indicated in the network outputs is then re-injected in the network inputs. This makes this model similar to the ANN model proposed by Jordan[2, 17], where the network outputs are re-injected in the inputs through a recurrence known as context units. Using these context units we provide the information about the current state of the network.

The neural network simulator adopted was the Stuttgart Neural Network Simulator - SNNS[4], it is a free software, and a quite complete neural network simulator that have several additional tools that allow us to create scripts and execute learning and simulation tasks in batch mode. The SNNS facilities also simplify the analysis of the obtained results and creation of graphic plots.

The neural input variables used were: the data from the six sonar sensors, the odometer and the current state of the parking process. In the network output are obtained the actuators activation commands (speed and steering wheel rotation), as well as the indication of the next state of the parking process. This network output information allows to simulate a FSA, controlling the

[4]SNNS – http://www-ra.informatik.uni-tuebingen.de/SNNS/

actuators and the FSA state, which can be changed according to the progress of the parking maneuver.

The states and actions were coded on a "1-of-N" basis in the learning database. The neural inputs and outputs were defined as follows:

- Inputs: current state (6 binary inputs); state of the sensors (6 numerical inputs normalized between 0 and 1); odometer (1 numerical input);
- Outputs: speed (3 binary outputs: forward, backward and stopped); steering wheel angle (3 binary outputs: Turn to the left, Straight forward and Turn to the right); next state (6 binary outputs).

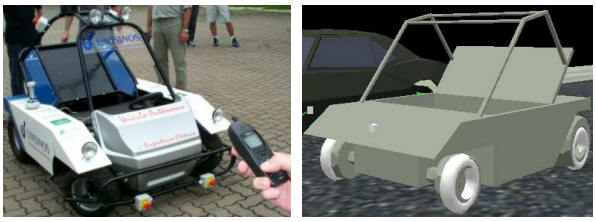
We point out that to know the current state is very important to allow a correct interpretation of the sensors (same situation, different actions), for example, when searching for a free parking space, if lateral sensors indicate the proximity of parked cars then this indicates that we should continue to move forward until a parking space is found, but when entering in a parking space after the vehicle was positioned outside of it, if the lateral sensors indicate the proximity of a parked car (exactly as in the previous situation) then this indicates that we should start to move backward and turn the steering wheel to the right.

So, correct state transitions are very important to accomplish the autonomous parking task. In our experiments we noticed that the neural net was perfectly capable to learn state transitions (next state outputs were correctly generated). Once the neural learning phase is finished, it is capable of accomplishing the vehicle parking without any human intervention.

4 Implementation

The SimRob3D simulation tools[8, 9] were used to create the virtual environment and to interface the vehicle devices with the SEVA3D autonomous controller implementation. The virtual environment was modeled using the 3DStudioMax software, creating 3D models of the road, the parked cars and our autonomous vehicle.

The model of the vehicle used to accomplish the parking task is a reproduction of a real Mini-Baja Buggy available in our research laboratory. This vehicle was developed by the GPVA Research Group at Unisinos. The real vehicle was automated and now it can be controlled from remote devices, like a cell phone (see available videos in the GPVA web site). The Figure 6 shows the actual vehicle used as model of the virtual autonomous vehicle. At the present time we are working on the real vehicle instrumentation, adding sonar sensors, and soon we plan to test the SEVA3D controller with this real vehicle.



(a) Real vehicle (b) Simulated vehicle

Figure 6. Automated mini-baja vehicle

5 Experimental results

The learning database was created with 5000 examples, (2500 for the learning and 2500 for the generalization test), each one with 13 inputs and 12 outputs. The results obtained in the ANN learning, considering the average of 10 simulations with different initializations, were a learning performance with a mean score of 96.83% correct answers. A correct answer is when all outputs signals are corrected, considering a Score Thereshold of 0.4. Three neurons were used in the hidden layer of the neural network, and the mean number of epochs was 192 (best epoch of generalization). The Figure 7 shows a exemple of parking manoeuvre[5].

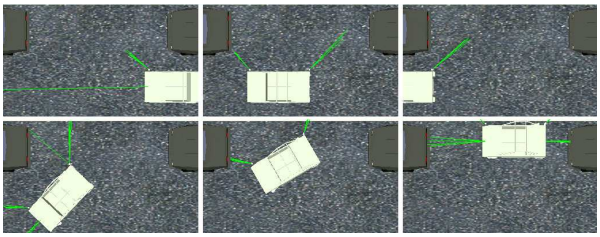


Figure 7. Parking maneuver

The examples that generated an incorrect network output were individually analyzed and in all cases we noticed that the wrong answers don't cause problems in the FSA state transitions. The trained ANN was perfectly capable to change from the current FSA state to the next FSA state with no errors. This is very important once correct FSA state transitions are critical to obtain a successful parking maneuver. In the vehicle pull out, the mean of the learning was 98.43% with the same neural network architeture. This demonstrates that the implemented system is safe and robust to control vehicles in parallel parking and pull out tasks execution.

[5]Some videos demonstrating the parking manoeuvre are available in http://inf.unisinos.br/˜osorio/seva3d/

6 Conclusions and perspectives

This work main goal was to develop a simulator for autonomous control of vehicles in parallel parking tasks. The proposed system should be able to create a realistic model of the real world application, so we implemented a simulation tool situated in a 3D environment, the SEVA3D. This system includes a 3D model of the vehicles and obstacles, and it also includes a 3D model of sonar sensors. The experimental results, accomplished with SEVA3D-A (control based on a FSA) and SEVA3D-N (control based on an ANN), demonstrated that our control system possesses the capacity to correctly control the vehicle. The main objective of the control system was achieved with success: to park vehicles in an autonomous way, not colliding against any obstacle present in the environment. In order to validate the SEVA3D vehicle control module, several experiments were accomplished with visual and numerical evaluations. The SEVA3D system experiments allowed to verify that the vehicle was correctly controlled in different situations, demonstrating that the proposed method is stable, safe and robust.

Although the experimental results were very good, we are still planning to improve SEVA3D/SimRob3D simulation model. The implementation of a new version of SEVA3D is being considered in order to include a more realistic physical simulation tool. We are considering to add rigid body dynamics simulation extensions, allowing to simulate force, torque, friction, gravity, terrain slopes, etc. In the near future we plan to implement SEVA3D-N in a real vehicle, an automated mini-Baja Buggy available in our research laboratory. The SEVA3D neural network will be adapted (trained) to use the new hardware (real sonar sensors) and the system will be evaluated in real world conditions.

References

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

[2] A. d. P. Braga, T. B. Ludermir, and A. C. P. d. L. F. Carvalho. *Redes Neurais Artificiais - Teoria e Aplicações*. LTC Editora, Rio de Janeiro, Brazil, 2000.

[3] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge Univ. Press, Cambridge, UK, 2000.

[4] J. D. Foley. *Introduction to Computer Graphics*. Addison-Wesley, xxviii, 1994.

[5] P. Garnier, T. Fraichard, C. Laugier, I. Paromtchik, and A. Scheuer. Motion autonomy through sensor-guided manoeuvres. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Kyongju, Korea, Sept. 1999.

[6] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, Upper Saddle River, NJ, 2 ed, 1999.

[7] F. J. Heinen. *Robótica Autônoma: Integração entre Planificação e Comportamento Reativo*. UNISINOS Editora, São Leopoldo, RS, Brazil, Nov. 1999.

[8] F. J. Heinen. Sistema de controle híbrido para robôs moveis autônomos. Master's thesis - applied computing, Universidade do Vale do Rio dos Sinos (UNISINOS), São Leopoldo, RS, Brazil, 2002.

[9] F. J. Heinen and F. S. Osório. HyCAR - a robust hybrid control architecture for autonomous robots. In *Proc. Hybrid Intelligent Systems (HIS)*, volume 87, pages 830–840, Santiago, Chile, 2002. IOS Press.

[10] M. R. Heinen, F. S. Osório, and F. J. Heinen. Estacionamento de um veículo de forma autônoma simulado em um ambiente tridimensional realístico. In *Anais do XIV Seminco*, pages 56–65, Blumenau, SC, Brazil, Oct. 2005. FURB Editora.

[11] M. R. Heinen, F. S. Osório, F. J. Heinen, and C. Kelber. Estacionamento de um veículo de forma autônoma utilizando redes neurais artificiais. In *Encontro de Robótica Inteligente (EnRI)*, Campo grande, MS, July 2006.

[12] M. R. Heinen, F. S. Osório, F. J. Heinen, and C. Kelber. SEVA3D: Using artificial neural networks to autonomous vehicle parking control. In *Proc. IEEE Int. Joint Conf. Neural Networks (IJCNN)*, Vancouver, Canada, July 2006.

[13] M. R. Heinen, F. S. Osório, F. J. Heinen, and C. Kelber. Uso de realidade virtual no desenvolvimento de um sistema de controle do estacionamento de veículos autônomos. In *Proc. of VIII Symposium on Virtual Reality*, pages 245–256, Belém, PA, Brazil, May 2006. Editora CESUPA.

[14] C. Kelber, C. R. Jung, F. S. Osório, and F. J. Heinen. Electrical drives in intelligent vehicles: Basis for active driver assistance systems. In *Proc. IEEE Int. Symposium on Industrial Electronics (ISIE)*, volume 4, pages 1623–1628, Dubrovnik, Croatia, 2005.

[15] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[16] C. Laugier, T. Fraichard, I. E. Paromtchik, and P. Garnier. Sensor based control architecture for a car-like vehicle. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 165–185, Canada, 1998.

[17] J. Ludik, W. Prins, K. Meert, and T. Catfolis. A comparative study of fully and partially recurrent networks. In *Proc. IEEE Int. Conf. Neural Networks (ICNN)*, volume 1, pages 292–297, Houston, TX, 1997.

[18] I. E. Paromtchik and C. Laugier. Autonomous parallel parking of a nonholonomic vehicle. In *Proc. IEEE Int. Symposium on Intelligent Vehicles (IV)*, pages 13–18, Tokyo, Japan, Sept. 1996.

[19] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. IEEE Int. Conf. Neural Networks (ICNN)*, pages 586–591, San Francisco, CA, Mar. 1993.

[20] D. Rumelhart, G. Hinton, and R. Williams. *Learning Internal Representations by Error Propagation*. MIT Press, Cambridge, MA, 1986.