Chapter 3
Simultaneous Navigation and Mapping in an Autonomous Vehicle Based on Fuzzy Logic

Álvaro Luiz Sordi Filho, Leonardo Presoto de Oliveira, André Schneider de Oliveira, João Alberto Fabro and Marco Aurélio Wehrmeister

This research presents the navigation control and mapping of an autonomous car by fuzzy logic that enables automatic obstacle avoidance in unknown environments. The strategy is based on a map of the environment, which is created according to navigation, to plan the trajectories avoiding obstacles through the search algorithm A*. The proposed approach is evaluated in a virtual environment, where the autonomous car should move among different obstacles.

3.1 Introduction

Autonomy is the capability of a vehicle (or robot) to move around a known environment, partially known or unknown, based on the perceptions of the environment(sensing), by building the map (mapping), making it possible to plan and replan the routes to the destination point, maneuvering around the obstacles without any interference from an outer source.

Á.L. Sordi Filho (✉) · L.P. de Oliveira · A.S. de Oliveira · J.A. Fabro · M.A. Wehrmeister
Graduate Program on Applied Computing (PPGCA), Federal University of Technology-Paraná (UTFPR), Curitiba, Brazil
e-mail: alalvaro_7@hotmail.com

L.P. de Oliveira
e-mail: leonardopoliveira@hotmail.com

A.S. de Oliveira
e-mail: andreoliveira@utfpr.edu.br

J.A. Fabro
e-mail: fabro@utfpr.edu.br

M.A. Wehrmeister
e-mail: wehrmeister@utfpr.edu.br

N. Nedjah et al. (eds.), *Designing with Computational Intelligence*,
Studies in Computational Intelligence 664,
DOI 10.1007/978-3-319-44735-3_3

The fuzzy logic is a tool used in the development of control strategies that allows a higher level of flexibility in the rules of the controller, and so, allows the implementation of the autonomous capability. The operation mode of a fuzzy system allows different approaches, like in [1], where the implementation of a fuzzy logic is used in reconfigurable systems, used in the speed in a vehicle's cruise control. Another similar approach is found in [2]. The fuzzy logic can also be used to control the motors in an electric vehicle as showed in [3].

Strategies for ADAS (advanced driving assistant systems) are also a reasonable application for fuzzy systems. In [4], a Fuzzy approach to adjust a PID control is discussed, to guide the vehicle to keep inside the lane. In [5], approach for collision avoidance for autonomous vehicle is taken. In [6] fuzzy systems are used to control the speed of a vehicle's cruise control, with online learning.

The navigation in a dynamic environment, as roads and highways, is a complex task due the amount of obstacles and environment changes that may happen (failures in the road, lack of traffic signs, inter-vehicle interaction, animals, pedestrian crossing, etc). The application of fuzzy systems is advised to situations like these, in [7], for example, a method to navigate in unknown environments based on different type of sensors is introduced. In [8], fuzzy systems are utilized in a autonomous vehicle's distributed control system.

This work focus on the capability of a vehicle to be autonomous, which means nothing more than an agent that is able to extract the information of an environment and use it to move around such environment in an efficient manner. The possible applications for these robots is huge, they can be used in the industry (AGV - autonomous guided vehicles), in the military (UAV - unguided autonomous vehicle), exploration, among others [9].

This work proposes a fuzzy system to control the navigation of an autonomous vehicle inserted in and static, partially observable (the sensing system cannot access all the environment information at all times), deterministic, discreet, and single agent. The validation experiments are developed in a simulation environment, and for these, a realistic vehicle model with sensors and actuators available in the market were used.

In the following sections, the concepts and the techniques used in the simulation are going to be discussed (Sects. 3.2, 3.3 and 3.4), the methodology adopted in the development (Sect. 3.5), the results and the final conclusions (Sects. 3.6 and 3.7).

3.2 Virtual Robot Experimentation Platform

The VREP (virtual robot experimentation platform) has resources to create, compose, and simulate robots. It has verification systems and can monitor remotely the actions performed by the robot under review [VREP 2015]. The VREP calculation module can determine the optimal parameters of a mobile joint to achieve the correct positioning, quickly calculates the possibility of a collision, allows planning a way to run in finite space, operates and interacts with programmed mechanisms and scene objects.

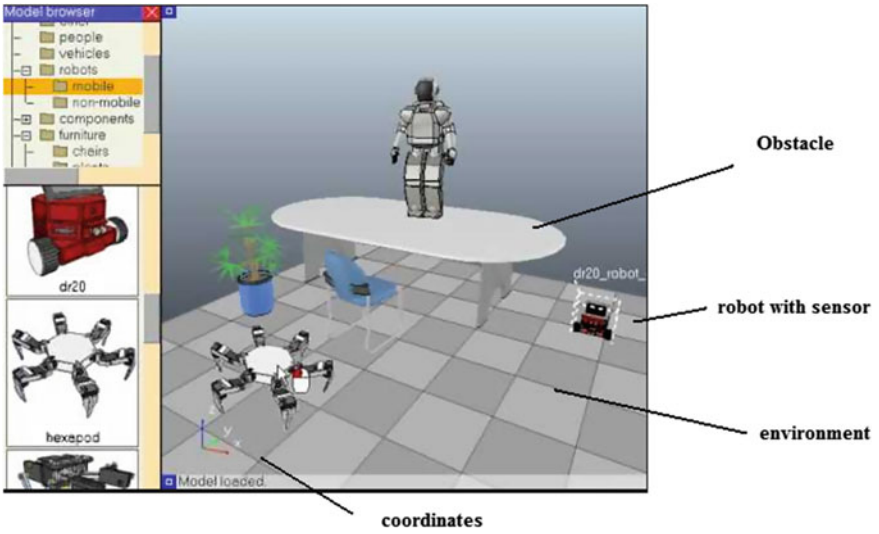

Fig. 3.1 Software VREP

The scene's objects can have cameras and lights, proximity sensors, force sensors, image interpretation cameras, and paths defined on 2D and 3D graphics. The simulation can be started, paused, and stopped at any instant of time and can be performed in real time or by approximate running time. Figure 3.1 illustrates the simulation environment VREP and scene's objects.

The self-guided car was imported and prepared from the robot library available in VREP. Environmental obstacles were inserted with properties "collidable," "detectable," and "measurable". The choice of the path to be traveled by the vehicle considered a starting point, the properties of the chosen path, coordinate or objective to achieve (meaning the goal location). Data was also inserted in the environment as the minimum and maximum detection distance and the minimum diameter of carriage movement.

The VREP has the application programming interface (API) that lets you access your library and services available through another programming language or application. It is possible to use C/C++, Java, Python, or mathematical software Matlab.

3.3 Autonomous Vehicle

Autonomous vehicles can have different topologies, which are dependent mainly of the mobility and maneuvering necessary for the application. An autonomous car can be considered as a mobile robot with different sensing systems, used in the perception of the environment. This robot is also able to move without the interference of an

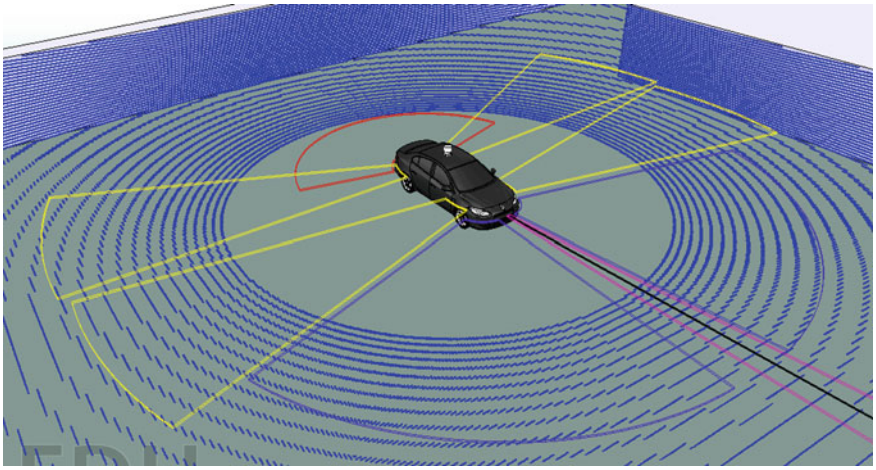

Fig. 3.2 Renault Fluence autonomous vehicle, and range of the sensing systems

extern operator. This vehicle uses a drive system called Ackerman geometry, which hold four wheels, of which two are fixed and the other two are directional. A further discussion about this geometry is available in [10].

The environment perception is acquired by sensors that measure the environmental quantities. Autonomous vehicle, typically, use sensors to identify objects (cameras, radars, LIDAR-light detection, and ranging, among others). In this context, the current work utilizes a regular sedan vehicle, more specifically a Renault Fluence, which was modeled in a virtual experimentation system V-REP (Virtual Robot Experimentation Platform), of Coppelia Robotics [11]. In the vehicle, were placed four radar systems (ultrasound) of medium range to identify possible objects on the sides (typically used to lane change assistance, and automated parking). At the front, a long range radar (ultrasound) was placed to detect object even in high speed. At the rear, parking sensors (ultrasound) was placed and also a LIDAR sensor (Hukuyo, a laser sensor) to help in the environment perception, as can be seen in Fig. 3.2.

3.4 Navigation and Mapping

The Renault Fluence's autonomy is achieved with a group of techniques called SLAM (simultaneous localization and mapping), which allows that the building of the map itself, is done during the robot's navigation (a further discussion about this approach is presented in [12]).

SLAM is one of the most addressed methods in the robotic area. This technique refers to how the robot is going to behave in a unknown environment and how it

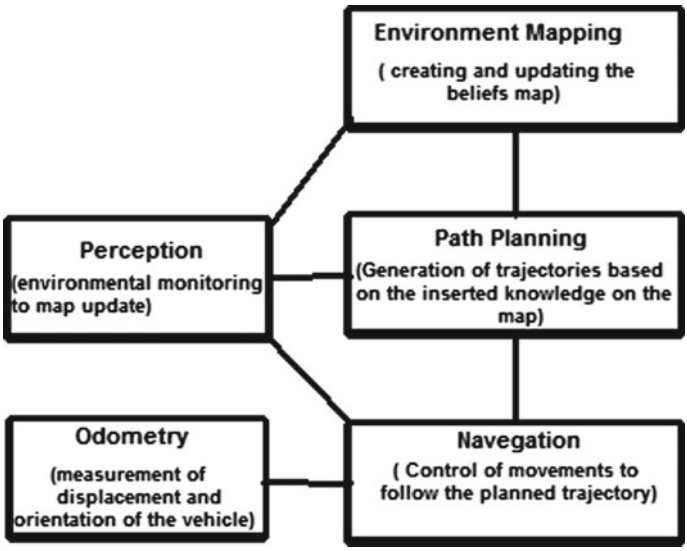

Fig. 3.3 Simultaneous localization and mapping structure (SLAM)

is going to develop the knowledge, while it moves around in the environment. The environment perception is performed by sensors that can be classified accordingly to its measurement as: proprioceptives (measure the greatness of the vehicle's inside) or exteroceptives (measure environmental greatness). They can also be classified as: passive (measure the environment energy) or active (they emit their own energy) [12]. There sensors detect the information that the robot needs to create its own navigation map. Figure 3.3 represents the inner structure of a SLAM system.

Sensors capture the perceptions of the environment and send them to the Mapping mechanism, which is responsible to create and update the knowledge map. The environment information allows the robot to locate itself (inside the environment), and then deliberate on how to act (navigate). This structure can also predicts the presence of an object that is not currently present in the environment.

The constant updates on the map is a basic need of the SLAM technique, because as the robot is moving around the map and detecting new obstacles on the map, in a second pass through the same position the object might not be there anymore. Is this case, if a static map was used, the robot will act as if the object was still there, resulting in errors.

The creation of the knowledge map uses a data structure called costmap, in which the robot define the cost of each point in the map. The costmap is used to calculate the route to the destination (trajectory planning), which is going to be used as reference to the navigation controller. The trajectory is calculated using the costmap, looking for the path with the smallest cost [13].

The information added to the map have error in both position and size, due to the face of measurement errors, which directly interfere in the navigation and might lead

to collisions. Therefore, every object present in the map has a shadowed boundary, with costs smaller than the obstacle itself, which lower the probability of the robot choosing that region on its route. This a fundamental characteristic in situations where the vehicle must calculate if it can move through a narrow pass.

3.4.1 Trajectory Planning

According to Delling [14], the A* algorithm search for and finds if possible, the path with the smallest cost from a start to a goal node (of one or more possible goals). To achieve this A* algorithm move through a graph (map) and follows the way with the lower expected cost, holding a priority ordered queue, in which each elements represents a piece of an alternate route along the way.

The A* algorithm uses a cost function of a node n (usually called f(n)), which is defined by the cost already known, g(n), added to the cost of the estimated path, h(n) (h for heuristic), to choose in which order each node is going to be visited in the tree. This cost function is the sum of two functions: $f(n) = g(n) + h(n)$.

Delling [14] also says that h(n) must be na admissible heuristic, it should not overestimate the cost for the destination. Therefore, for the application that must calculate the distance between two points, h(n) can represent the straight line between start and objective, because this is always the smallest distance between two points or nodes. This is also called the Euclidean distance.

If the heuristic h satisfies the additional condition $h(x) = d(x, y) + h(y)$ for each edge (x, y) of the graph (where d indicates the length of edge), h is the called consistent. In this case, the A* algorithm can be implemented more efficiently, roughly, no node needs to be processed more than once and A* is equivalent to running Dijkstra's algorithm with reduced cost $d'(x, y) = d(x, y) - h(x) + h(y)$ [10].

The A* algorithm is responsible for allowing the autonomous vehicle plan the optimal path between the point that it is up to the chosen goal. An adjustment was made in the A* algorithm that instead of returning all points of the optimal path, the A* algorithm returns only the conversion points present in the optimal path. It was determined that conversion point is a point at which the car should make a turn, the distance between two turning point should be done in a straight line. For example, if the car is in the space (0, 0) (x, y); and must get to the point (10, 10). The A* algorithm can return the following way: (0.1) (0.2) (0.3) (0.4) (0.5) (0.6) (0.7) (0.8) (0.9) (0.10) (1.10) (2.10) (2.10) (4.10) (5.10) (6.10) (7.10) (8.10) (9.10) (10.10); in all are 20 points. With the adjustment made the algorithm returns only (0, 0), (0, 10), (10, 10). In this way, you can determine a straight line between the points (0,0) and (0.10), and another straight line between the points (0.10), (10:10), and this will be the path that the car will follow.

3.4.2 Navigation

Fuzzy systems are based on fuzzy logic, which has different characteristics from traditional logic. A traditional logical proposition has two specific situations: it is completely true or entirely false. But in fuzzy logic, a premise can vary between true or false, that is, instead of being only values 0 or 1, can occur in a premise assume a value between 0 and 1. As can happen to a premise be partly true or false [15].

The process of a fuzzy system is based on fuzzy sets, the membership functions, and the fuzzy rules. The sets are partitions of all the possible values for the input variables. The membership functions define the fuzzy sets. The rules are used in the system inference engine. They use complementary operators, union, and intersection to establish the relationship between the input variables and system output [Coppin 2013].

The partition of a fuzzy set assigns degrees of relevance to the elements of this set. The data used to create the rules were generated based on tests done on the platform. Aspects were tested as braking and acceleration time of the simulated vehicle. From these tests it was created a data table, which served as the entrance to the creation of inference rules. To create the fuzzy inference rules were used the concepts of Wang Mendel algorithm [16].

This Wang Mendel algorithm is used for the automatic extraction of the relevant rules to a fuzzy system and can be summarized in the following steps: Define the number of linguistic terms and partition the universe of all the input variables; Building a fuzzy rule for each member of the set of training points—For each input variable, select the higher level membership function; Calculate the degree of activation of all rules using an appropriate operator.

In the autonomous vehicle, the fuzzy system is designed to control not only the car's speed, but also control the rotation of the wheels when it is needed. In all three are applied Fuzzy controllers.

The first fuzzy system consists of two inputs (distance and speed of the front wheels) and an output (speed). The goal of this fuzzy system is to ensure that the car do not collide with the obstacle. Figure 3.4 illustrates this system. The variable "distance" ranges between 5 and 20 m, the "rotation" variable ranges between −40° (left oriented) and 40° (right oriented) and the output ranges between 15 and 40 km/h. Another simple control system complements the operation and ensures that if the car is 0.5 m from the obstacle, a reverse maneuver is performed, since from that position this is the only alternative.

When the car is at a shorter distance than 5 m, the control is based on the front sensors and causes the car to turn to the side closest to the desired point. This control determines the difference between the direction the car is and the direction it should be to follow the optimal path. The output is the angle that the car should turn to follow the desired path and varies from 180 (left) to 180 (right), as illustrated in Fig. 3.5.

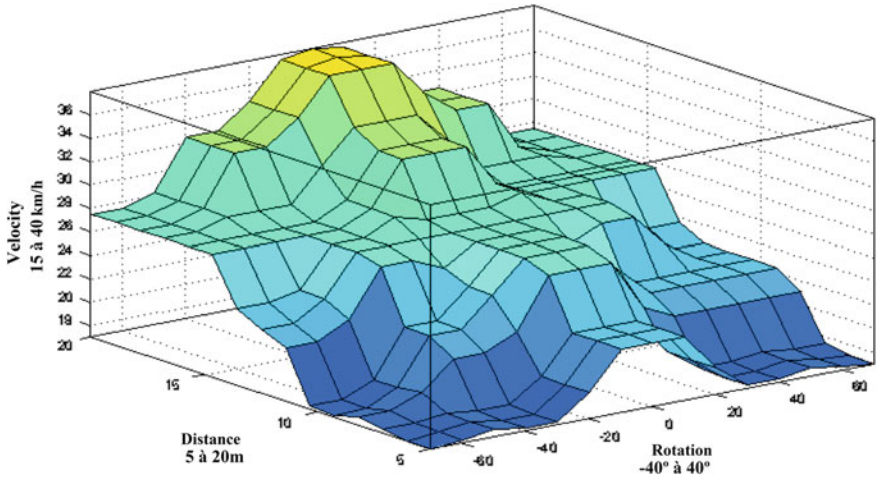

Fig. 3.4 Auxiliary control system that helps to avoid car collision with environmental obstacles

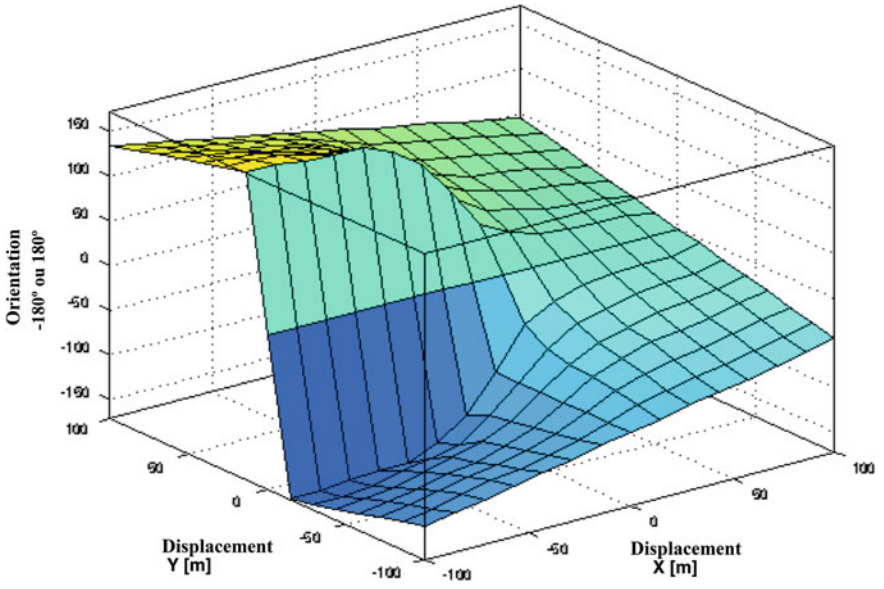

Fig. 3.5 Control system that determines the difference between the car's direction and orientation of the goal point

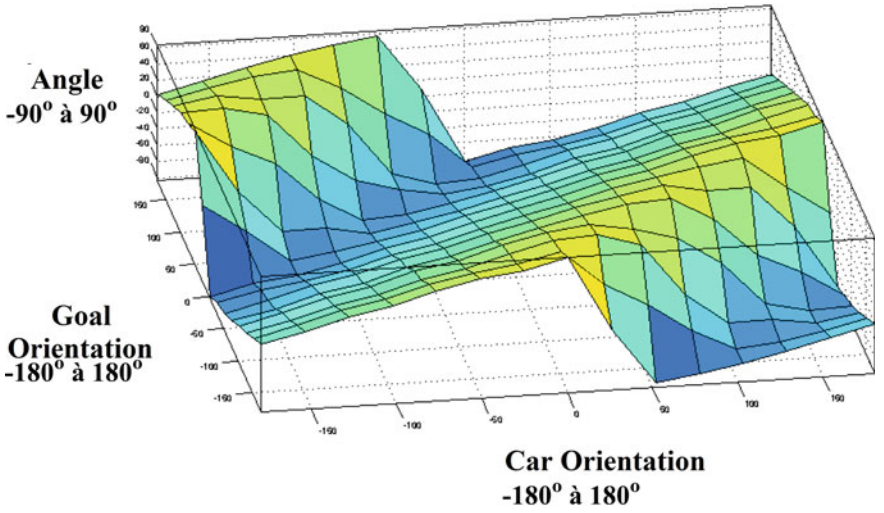

Fig. 3.6 Fuzzy system to supervise the car turning, so that it reaches the correct orientation to reach the goal

The third fuzzy control system (Fig. 3.6) oversees the car turning and determines both rotation of the wheels, and the speed of movement. This control is important because if the car is in a narrow place, the maneuver cannot be performed in a single movement.

3.4.3 Decision Tree

Decision tree is the simplest form among the most used decision models, but is quite effective. Among its main strengths, its transparency and the ease of developing are highlighted.

The decision tree structure is very similar to the if-then structure, widely used in expert systems and rating systems [17].

At the entrance of a decision tree are received attributes that can be continuous or discrete, then the tree will reach a final decision based on their tests. In the tree structure, each node represents a different test and each leaf node of this tree represents a value that is returned if this leaf node is reached.

In the decision tree, each node is the knowledge of the expert leading the search for one of its child nodes. So as tou move down the tree, the desired system configuration will be selected and thus choosing the desired behavior [18].

For the autonomous vehicle, the decision tree is used to select which group of actions the agent must take, i.e., to decide if it should just follow the path determined by the trajectory planner, dodge an obstacle, or to stop and make a reverse in case of the available space is not sufficient to maneuver.

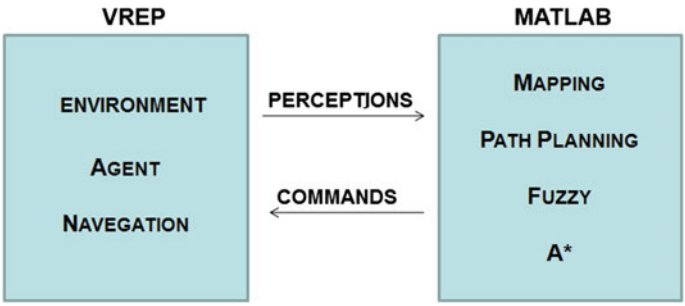

Fig. 3.7 Architecture simulation

3.5 Simulation

Figure 3.7 presents the architecture of the simulation to be developed.

The VREP is responsible for simulating the environment, the agent (vehicle), and the navigation positioning information (which is calculated by MATLAB). Agent perceptions (signals picked up by the sensors) are sent from VREP to MATLAB and this in turn is responsible for performing the trajectory calculations (A*), determine the correct speed that the car should be in accordance with the current situation (Fuzzy), in addition to building the mapping and planning trajectories. MATLAB then returns the commands, which are these trajectory and velocity information that the vehicle must follow.

For the simulation to work correctly, there must be integration of code developed in MATLAB, with the components created in VREP. Communication between the two software is executed via socket, and to facilitate this, in this paper we chose to use the API developed by Coppelia (company responsible for VREP).

The modeling agents is described in more detail in the online tutorial VREP; however, it is required that each vehicle component is declared as a clear object name, since to access these components via MATLAB the names that were defined in VREP are used as unique identifiers.

For example, when you want to turn the front wheel 30° right, the following command is sent.

- vrep.simxSetJointTargetPosition(clientID, RhtWheelHandle, −degtorad(30), vrep.simx_opmode_oneshot);

The vrep.simxSetJointTargetPosition command is used to rotate a particular component present in VREP. Arguments are, respectively, clientID − name of the simulation; RhtWheelHandle − the name given to the right front wheel; −degtorad(30), which converts 30° to its equivalent value in radians, and vrep.simx_opmode_oneshot − representing that this command must be run only once. In this example, it is clear that the choice of name facilitated in performing the function, as RhtWheel can be easily associated with the front right wheel.

3.6 Case Study

The case study is a Renault Fluence (Fig. 3.8) vehicle, which is placed in a virtual environment with many obstacles. As defined by the Ackerman geometry, only the front wheels determine the direction, i.e., the rear wheels are free. When the car needs to make a turn, only the front wheels must be acted on. The car's size is about 4 m long and 2 m wide.

The environment is a closed area of 100 m × 100 m, as illustrated in Fig. 3.9. In this space were defined spaces by which the vehicle can follow and also obstacles that it should divert. Altogether there are four obstacles in the shaped of cubes of varying sizes. Besides the obstacles, there are walls which limit the passage of the vehicle.

The side tracks (bounded by walls) are narrow to hinder a possible curve the car must do. This difficulty requires the control to be more efficient because, depending on the speed it is not possible that the car maneuver without having to move in the opposite direction (reverse) or maneuver itself to fir the curve.

The experiment starts with the car stopped at a certain point and the map of empty knowledge. At that moment, the objective point is designated for the autonomous vehicle. The first task is to carry out the planning of the trajectory based on the knowledge map, then the path is calculated by A* and then when the algorithm returns the "sub-goals" (turning points) the car starts moving at the resulted path. The fuzzy controllers are responsible for making the car follow the planned trajectory.

Fig. 3.8 Autonomous vehicle—virutal Renault Fluence

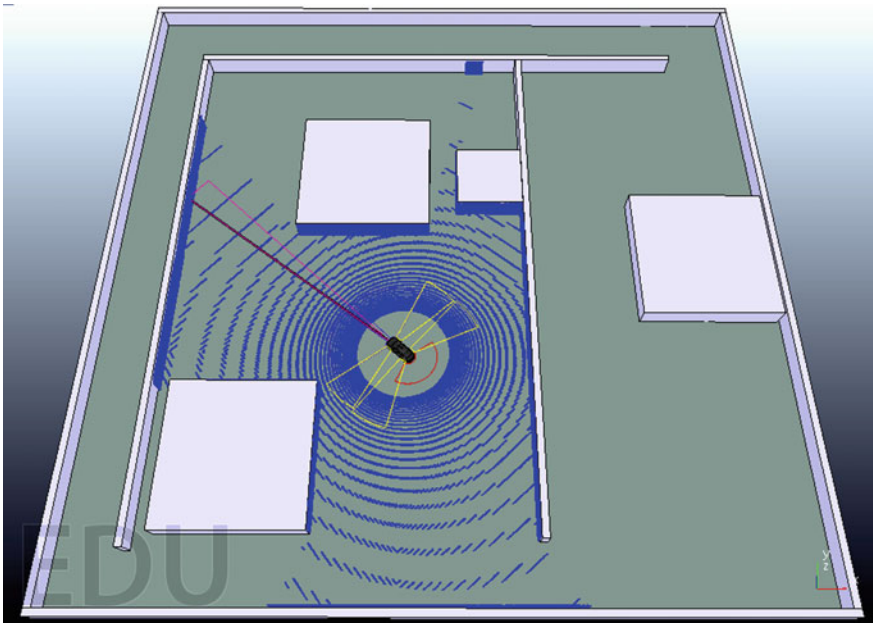

Fig. 3.9 A car that interact in this environment, which is unknown at first (but is being mapped), whose object reaches a certain point, traversing the optimum path (derived from the A* algorithm)

The car follows the optimum path at the same time it updates the environment map. When one eventually is very near or in the planned trajectory, the fuzzy control action operates on the front sensor for the vehicle to deviate from obstacles.

Figure 3.10 depicts the map of the environment being built as the car moves. The brighter areas represents he position where the obstacle was encountered, while the area in gray is the inflation that the car has to calculate around obstacles.

When the obstacle is overcome, the control indicates that the car should go back to follow the path determined by the trajectory planner, but the car may not be oriented in a way that is must only move forward. At this point, another fuzzy controller is activated to correct the orientation of the vehicle according to the desired trajectory (determined by the trajectory planner).

After adjusting the orientation of the vehicle to the optimum path, the car return to the state of following the path determined by the trajectory planner to reach its goal. Due to the car's size, it was determined that the goal is any point in a distance of 1 m from the goal point (tolerance range). This measure was taken to prevent the vehicle to keep making small maneuvers to be exactly at the desired point, since these maneuvers took a long time and are negligible, since the car was already above the point and was just trying to align the car's center with the goal point.

Fig. 3.10 Environment partially mapped by the vehicles with help of the perception system

The fuzzy controller for steering correction, used interactively, enables the direction (orientation) fine tuning. The fuzzy controller responsible for the vehicle's speed depending on the distance of the obstacles was important to ensure that the car does not collide with objects.

The correction in the direction of the vehicle is an intrinsic difficulty to the problem, because as it is a regular vehicle, any direction adjustment requires continuous forward and backward movements, until it reaches the desired angle.

The vehicle was also tested in a different scenario. This time the scenario is a bit closer to a real-life situation. The car was placed inside a parking lot of a shopping mall. Figure 3.11 represents this scenario with an upper view. The whole scenario is composed by the shopping mall at the middle, the vehicle at the left upper corner and eight light posts, being 4 at the north and 4 at the south wall The textures of the shopping mall exterior and the parking lot floor were removed to improve performance of the simulation.

As the previous experiment, different positions were used as goal for the vehicle. When the car reached its destination, another point was set as goal. By switching destinations, it was nearly possible to map the entire parking lot with the shopping mall at the middle. Figure 3.12 contains the resulting map of the scene.

The places that were not mapped in Fig. 3.12 are spots where the vehicle did not get close enough so its sensors were able to identify the walls. This happened because during the path planning stage, the search algorithm was able to find a better path that did not go through the unmapped spots.

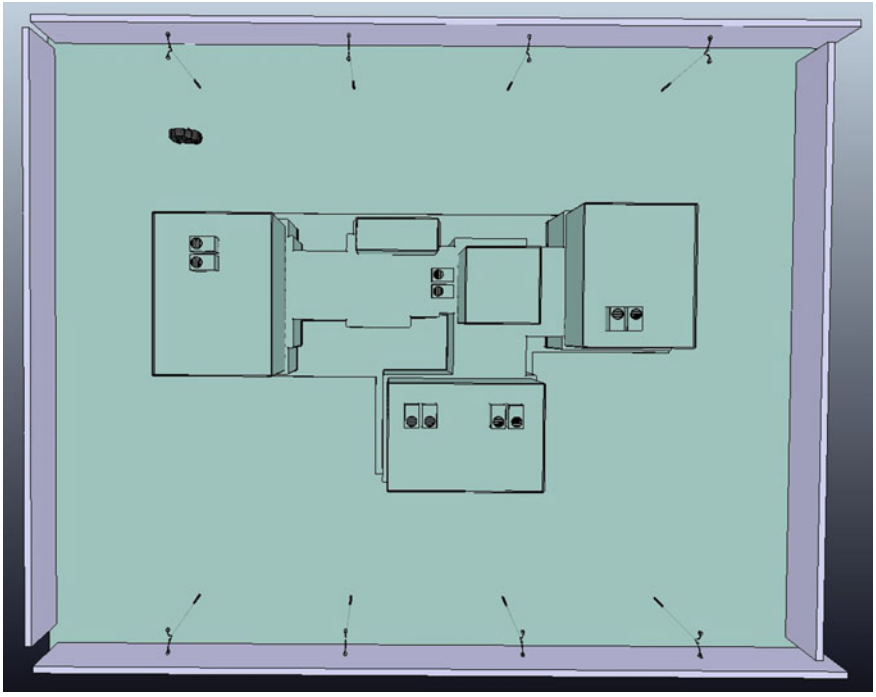

Fig. 3.11 Shopping mall and parking lot scenario

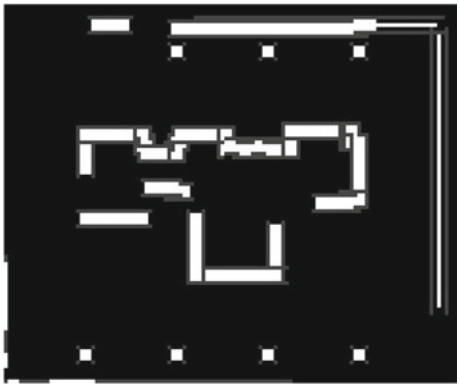

Fig. 3.12 Partial map of the shopping mall and parking lot scenario

3.7 Conclusions

The simulations show that the system responds effectively. Initially in a totally unfamiliar environment, with a goal position, the robot calculates the optimal route, this being a straight line. As the robot detects objects and updates its knowledge map, as soon as it finds an obstacle in its path the route is recalculated.

The method A* was used, despite producing the desired result, it requires relatively much computational time, causing the car, in the simulation, stand still while the trajectory planner find the optimal path. The fuzzy control systems (Fuzzy) have proven their efficiency in relation to what was expected. The guidance control to the next sub-goal returned valid responses to any situation where the car and the goal met.

The SLAM used in the work also proved to be satisfactory to map the environment. The detail to be highlighted at this point is that inflation in the mapping area can interfere decisively in the development of history. In the case of this work was necessary to increase the inflation area to ensure that the car does not collide with the corners in the environment. The autonomous vehicle is able to navigate in different environments, with smoother curves, and less narrow lanes.

In addition to different environments, it is suggested to implement a control to make the transition from the direction of the vehicle between softer sub-goals.

Acknowledgments To the Araucaria Foundation and the Renault of Brazil for partial funding from the research project.

References

1. Nedjah N, Sandres PRSS, de Macedo Mourelle L (2014) Customizable hardware design of fuzzy controllers applied to autonomous car driving. Expert Syst Appl 41(16):7046–7060
2. Petkovic D, Issa M, Pavlovic ND, Zentner L (2013) Intelligent rotational direction control of passive robotic joint with embedded sensors. Expert Syst Appl 40(4):1265–1273
3. Peng X, Zhe H, Guifang G, Gang X, Binggang C, Zengliang L (2011) Driving and control of torque for direct-wheel-driven electric vehicle with motors in serial. Expert Syst Appl 38(1):80–86
4. Wang Q, Xu S-Z, Xu H-L (2014) A fuzzy control based self-optimizing PID model for autonomous car following on highway. In: International conference on wireless communication and sensor network (WCSN), pp 395–399, 13–14 December 2014
5. Llorca DF, Milanes V, Alonso IP, Gavilan M, Daza IG, Perez J, Sotelo MA (2011) Autonomous pedestrian collision avoidance using a fuzzy steering controller. IEEE Trans Intell Transp Syst 12(2):390–401
6. Onieva E, Godoy J, Villagra J, Milanes V, Perez J (2013) On-line learning of a fuzzy controller for a precise vehicle cruise control system. Expert Syst Appl 40(4):1046–1053
7. Farooq U, Hasan KM, Amar M, Asad MU (2013) Design and implementation of fuzzy logic based autonomous car for navigation in unknown environments. In: International conference on informatics, electronics and vision (ICIEV), pp 1–7, 17–18 May 2013
8. Martinez-Barbera H, Herrero-Perez D (2014) Multilayer distributed intelligent control of an autonomous car. Transp Res Part C: Emerg Technol 39:94–112

9. Terano T, Asai K, Sugeno M (eds) (2014) Applied fuzzy systems. Academic Press, New York
10. Ackermann J, Bunte T, Odenthal D (1999) Advantages of active steering for vehicle dynamics control
11. Rohmer E, Singh SPN, Freese M (2013) V-REP: a versatile and scalable robot simulation framework. In: IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 1321–1326, 3–7 November 2013
12. Siegwart R, Nourbakhsh IR, Scaramuzza D (2011) Introduction to autonomous mobile robots. MIT Press, Cambridge
13. King J, Likhachev M (2009) Efficient cost computation in cost map planning for non-circular robots. In: IEEE/RSJ international conference on intelligent robots and systems. IROS. IEEE
14. Delling D, Sanders P, Schultes D, Wagner D (2009) Engineering route planning algorithms. Algorithmics of large and complex networks. Springer, New York, pp 117–139
15. Melo LG (2011) Sistema Fuzzy Probabilístico Geração Automática de regras e Defuzzificação Bayesiana. Dissertação de Mestrado em Informática Industrial. Universidade Tecnológica Federal do Paraná, Curitiba
16. Wang L-X, Mendel JM (1992) Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Trans Neural Netw 3(5):807–814
17. de Soárez PC, Soares MO, Novaes HMD (2014) Modelos de decisão para avaliações econômicas de tecnologias em saúde. Revista Ciência & Saúde Coletiva 19.10
18. Pozzer, CT (2006) Aprendizado por árvores de decisão. 2010, Universidade Federal de Santa Maria, Rio Grande do Sul