

# Car Navigation and Collision Avoidance System with Fuzzy Logic

Andri Riid, Dmitri Pahhomov and Ennu Rüstern  
Department of Computer Control  
Tallinn University of Technology  
Ehitajate tee 5, 19086 Tallinn, Estonia  
E-mail: andri@dcc.ttu.ee

**Abstract**—A navigation control and collision avoidance system for delivering a car to the arbitrarily positioned loading dock is designed, based on the fuzzy trajectory mapping unit (TMU). Simulated driving experiments in different environmental conditions demonstrate that the designed system shows good performance. Modular structure of the control system facilitates both efficient control knowledge acquisition (which is encapsulated in TMU) as well as further development of the control system to accomplish more demanding tasks.

## I. INTRODUCTION

In fuzzy control research, the cab part of Nguyen and Widrow's truck backer-upper [1] is often picked as a test object [2-8] for its nonlinearity on one hand and lack of traditional control system design methods that could be applied for this problem, on the other. At first glance, the control problem also seems one of these cases where the historical application area of fuzzy controllers - knowledge-based control - would be appropriate solution. This seems so because ability to drive a car is a very common skill among people thus it should not be too difficult to find an expert whose verbal instructions would then constitute the core of the control system. Car driving skill, however, is usually learned to a degree where it rarely intrudes on consciousness (the occasions when it does are unusual circumstances like a potential accident or a situation the driver is not used to (i.e. he/she has not yet learnt it). Consequently it is difficult to extract appropriate rules from an expert because of one's inability to explain how one accomplishes the task, and consequent difficulties in putting it down in terms of fuzzy logic. The design of knowledge-based controller therefore becomes much more difficult than was assumed in the first place.

In our previous study [9] we have shown how the decomposition of the control problem can substantially simplify and make very natural the most critical part of controller design - expert knowledge acquisition. This is because we become are focused on information concerning car optimal orientation in two-dimensional space (that is much easier to explain than the very actions on the steering

wheel), which ultimately results in an efficient solution of the problem. Moreover, the decomposition reduces dimensionality of the problem (which allows us to involve ourselves with more complex control goals such as knowledge-based control solution of the original truck backer-upper problem, as shown in [10]). The present paper further develops the approach taken in [9] and [10], by first generalizing functionality of the central unit of the control system, which allows us to specify an arbitrarily chosen loading dock position and then utilizing it for trajectory management in subsequently developed collision avoidance system.

## II. CONTROL OBJECT

The car position is determined by three state variables  $x$ ,  $y$  and  $\Phi = [-90^\circ, 270^\circ]$ , where the latter is the angle between truck's onward direction and the  $x$ -axis (Fig. 1). The width and length of the car are denoted by  $w$  and  $l$ , respectively ( $w = 2$ ,  $l = 4$ ).

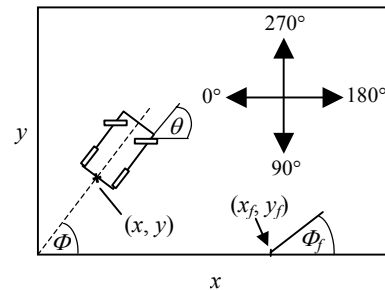


Fig. 1. Car and main variables

The problem is formulated as follows: the car must arrive from the arbitrary initial position  $(x_i, y_i, \Phi_i)$  to the predefined loading dock  $(x_f, y_f, \Phi_f)$ . Car moves forward or backward with the fixed speed (i.e. speed control is not our concern). To control the car, appropriate steering angle  $\theta = [-45^\circ, 45^\circ]$  must be provided.

Moreover, the car must be able to avoid contacts with the bounding walls (ranges of  $x$  and  $y$  can be freely specified) and with any other obstacles on its way.

For car movement simulation, the following set of equations is used

$$\begin{cases} \Phi_{t+1} = \Phi_t + (v\Delta t / l) \tan \theta \\ x_{t+1} = x_t + \Phi_{t+1} v \Delta t \cos \theta \\ y_{t+1} = y_t + \Phi_{t+1} v \Delta t \sin \theta \end{cases} \quad (1)$$

where  $v$  is the speed of the car.

### III. CONTROL SYSTEM

Implementation of the control task in the original control system in [9] is distributed between two units (Fig. 2). The main component of the system is the trajectory mapping unit (TMU) that specifies the optimal car angle ( $\Phi$ ) for the given point in input space determined by car current coordinates  $x$  and  $y$ . Computation of the actual steering angle  $\theta$  that would lead to the desired car orientation is carried out by a PD control loop (Fig. 2). The advantages of this configuration when compared to all-in-one approaches [2,3] become obvious. First, the number of input variables to the fuzzy system is reduced (which allows us to take  $y$  fully into account (usually neglected for the sake of simplicity in such contributions)). The overall structure of the control system also facilitates more efficient control knowledge acquisition and implementation because TMU encapsulates driving know-how that is quite obvious (see next subsection). Finally, feedback in the control loop stabilizes and enhances steering quality.

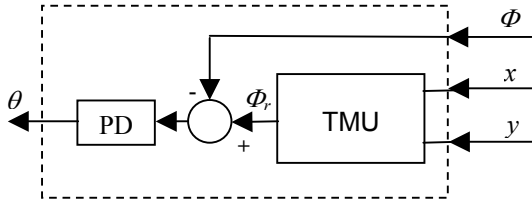


Fig. 2. Hierarchical control system.

In its current form, the control system, however, has its limitations. First, it has a rather small working range ( $x = [-25, 25]$ ,  $y = [0, 27]$ ). Secondly, if we, for some reason, want to specify the control goal different from  $[x_f = 0, y_f = 0, \Phi_f = 90^\circ]$  (which is the ultimate goal in [9]), it requires retuning of the whole TMU (obviously not a very good solution). The car is also ignorant of possible obstacles coming its way (including the bounding walls) and is therefore unable to avoid consequent collisions. The following subsections explain how these problems can be solved through the extension of the original control system.

#### A. Generalization of the trajectory mapping unit

Generalization of the TMU is accomplished in two steps. First, to extend the working range of the TMU, we use the

saturation functions for input variables,  $x$  and  $y$ , which limit the input values to the upper and lower limits of the TMU to ensure that car navigation will be governed by appropriate rules even when  $x$  and  $y$  appear to be outside the scope of TMU.

Moreover, it appears that on one hand, the original TMU rulebase (consisting of 28 rules) can be slightly optimized as some redundant rules (and membership functions) can be merged or removed (Fig. 3); on the other hand it has to be extended to cover the negative range of values of  $y$  where different actions have to be taken to bring the car home.

To avoid undesirable co-effects from the interpolation of antagonistic rules, the part of the TMU that corresponds to the range of negative values of  $y$ , is realized as a separate piece. (Fig. 4). All in all, this should solve our first problem - limited working range of the controller.

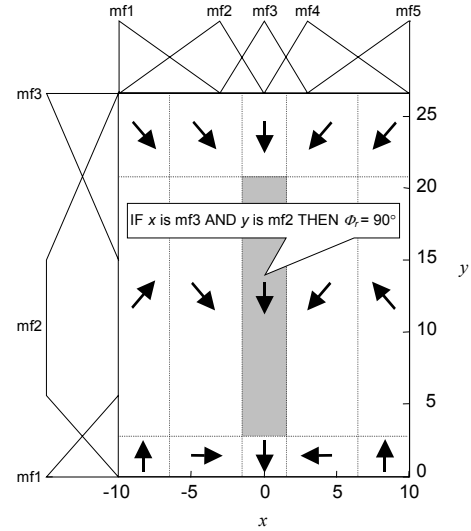


Fig. 3. 15 TMU rules responsible for trajectory management when  $y > 0$ . Each small arrow indicates optimal angle of the car corresponding to the given rule

In order to save ourselves from retuning the supervisor each time the final destination is changed, the original TMU in Fig. 2 is replaced by the one depicted in Fig. 5 that comes with the built-in interface between the actual values of variables  $x$ ,  $y$ ,  $\Phi$  and those that will be used for the computations in the interfaced TMU (which are denoted by  $x'$ ,  $y'$  and  $\Phi'$ ).

The values of  $x'$  and  $y'$  are obtained using the following formulas:

$$\begin{aligned} x' &= r \cos(90^\circ - \Phi_f + \Phi) \\ y' &= r \sin(90^\circ - \Phi_f + \Phi) \end{aligned} \quad (2)$$

where

$$r = \sqrt{(x - x_f)^2 + (y - y_f)^2} \quad (3)$$

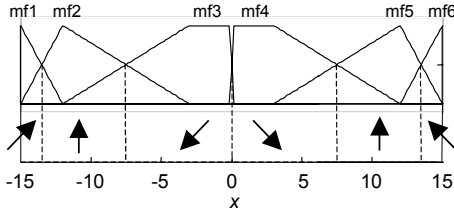


Fig. 4. TMU rules responsible for trajectory management when  $y < 0$ .

Set point value corresponding to the given destination  $(x_f, y_f, \Phi_f)$  is computed by

$$\Phi_r = \Phi_r' + \Phi_f - 90^\circ. \quad (5)$$

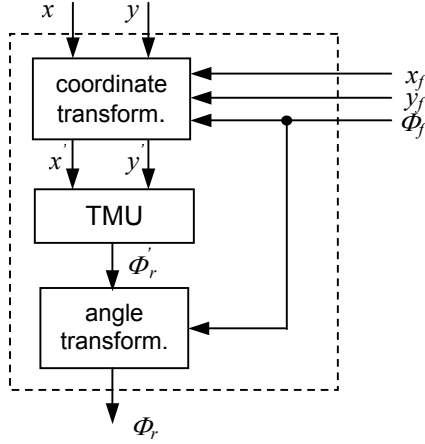


Fig. 5. Generalized TMU interface

Note that the expanded system can be used for car control in forward driving mode if we correct  $\Phi_r$  by  $180^\circ$  (the reason is obvious) and multiply the PD controller output  $\theta$  by  $-1$  (the latter is necessary because turning the wheels to the left would turn the car to the right in backward driving mode but would have the opposite result in forward driving mode)

Few simulations having initial conditions  $(-40, 85, 45^\circ)$  and  $(40, 20, 45^\circ)$  and dock locations specified as  $(0, 30, 0^\circ)$ ,  $(-25, 40, 260^\circ)$  and  $(30, 80, 45^\circ)$  show (Figs 6-7) that proposed control system is able to drive the truck home to arbitrarily chosen final destinations.

Note that the control system is currently optimized to the car characteristics specified in section 2. Although the control system is quite robust, substantial increase of car dimensions and/or smaller maximum steering angle would both hamper car maneuverability, which may result in a failure. If this is the case, however, it is possible to adjust TMU to the changes in car specification with the help of scaling factors applied to TMU inputs  $x$  and  $y$ .

### B. Collision avoidance system

For obstacle avoidance the car must be equipped with the devices that make possible object detection. 8 distance sensors are “installed” (although only four of them will be active at a time): left and right rear sensors  $s_1$  and  $s_2$ , right and left side sensors  $s_3 - s_6$  and left and right frontal sensors  $s_7$  and  $s_8$  giving readings  $d_1-d_8$ , respectively (Fig. 8).

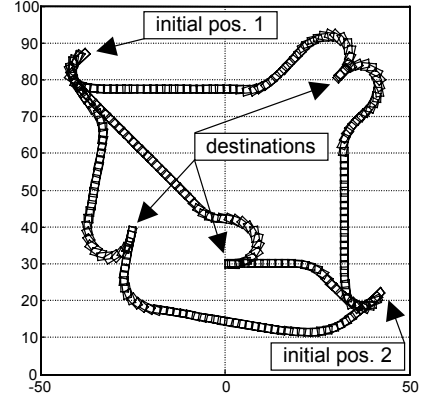


Fig. 6. Backward driving

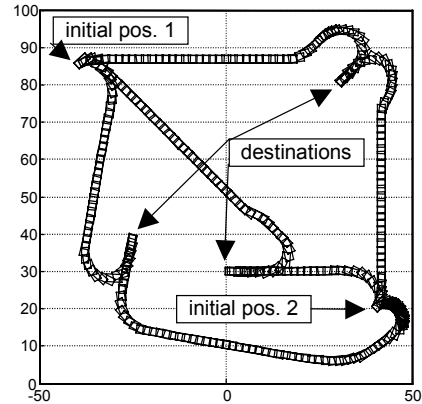


Fig. 7. Forward driving

Collision avoidance in backward driving mode works on the following principle: if an obstacle is detected in car’s way (either  $d_1$  or  $d_2$  is smaller than predetermined detection limit  $d_{det}$ ), first, the turning direction (left or right) is determined the on the basis of  $d_1-d_4$  (Fig. 9).

Based on the turning direction decision and orientation of the obstacle (that will be identified according to the rear distance sensor readings) a temporary gate is projected onto the bounding wall in car’s turning direction (Fig. 10). Note that presently, the turning direction selection is based on rear sensor readings, as there are no relevant obstacles in car side directions. Note that the opposite turning decision would result in the temporary gate projected onto the opposite (upper) bounding wall.

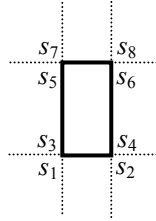


Fig. 8. Distance sensor placement scheme.

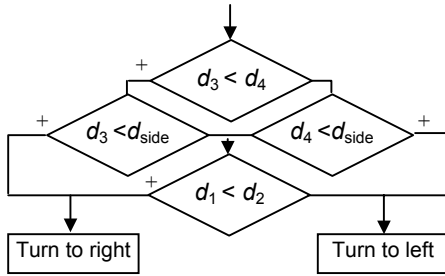


Fig. 9. Turning decision making

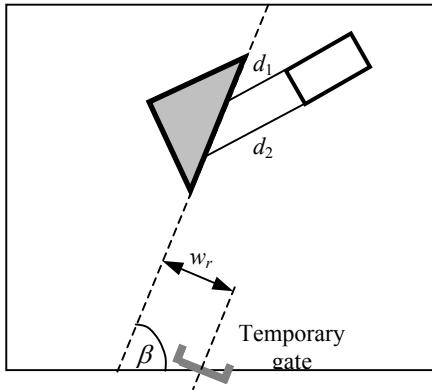


Fig. 10. Temporary driving goal

In forward driving mode the algorithm is basically the same, only we are using sensors  $s_5$ - $s_8$  instead of  $s_4$ ,  $s_3$ ,  $s_2$  and  $s_1$ , respectively and the obstacle orientation angle  $\beta$  must be corrected by 180 degrees.

Performance of the collision avoidance system is subject to the following parameters

$d_{det}$  – obstacle detection limit

$d_{excl}$  – rear sensor readings larger than  $d_{excl}$  will be emulated by  $d + w$ , where  $d$  is the rear sensor reading responsible for obstacle detection.

$d_{side}$  – determines if side sensor information will be used in turning direction determination (Fig. 9).

$w_r$  – the distance by which the temporary gate is shifted from the obstacle-projected line (Fig. 10).

### C. State transition control

We can speak of two states in car control – normally car control is carried out by the extended TMU described in section 3.1. (state 0) and when an obstacle is detected by

distance sensors, collision avoidance system (previous section) must take over the control (state 1) until it is safe to return to the normal state. For overall good performance state transition control must be smooth and effective. This is done in three steps as explained below.

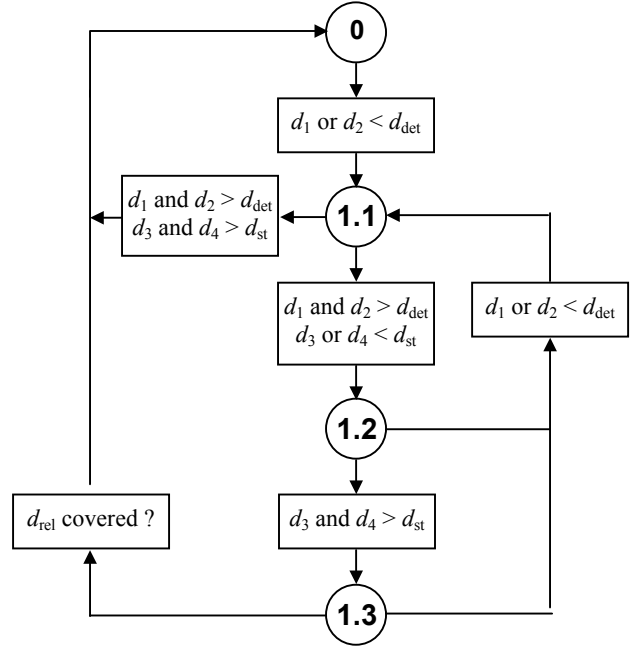


Fig. 11. State transition diagram (backward driving mode)

Collision avoidance system takes over if at least one of the sensors in the moving direction of the car registers distance smaller than  $d_{det}$  (transition from state 0 to 1.1 in Figs. 11 and 12). The car then starts to rotate because of the temporary driving goal that has been imposed in the moment of state transition and further state management will be delegated to side sensors if the obstacle comes to the reach of one of them (substate 1.2) - unless a new obstacle is detected and new temporary gate must be computed (return to substate 1.1). Eventual return to state 0 can principally happen if the obstacle is no longer present at car's side (substate 1.3). In certain circumstances, however, it may be necessary to delay the car's release further (too early release may cause a collision with the obstacle it currently passes), which is controlled by  $d_{rel}$  – small distance it has to cover while in substate 1.3.

State transition is independently regulated by two extra parameters ( $d_{min}$  and  $d_{max}$ ) that determine minimum and maximum time the system can spend in state 1, respectively. The aim of  $d_{min}$  is to make car control less sensitive to rapid changes of external conditions. The need for  $d_{max}$  becomes apparent when a bounding wall or similarly shaped connected obstacles happen to be in car's way because in this case the car cannot break out from state 1 on its own, even if it would be good for the overall goal.

#### IV. RESULTS

For the overall testing of the constructed control system in backward and forward driving mode two different environments have been designed – one presents obstacles of various shapes (Fig. 15) and the second one is a labyrinth (Fig. 16). In order to perform the requested tasks we need to determine proper settings of  $d_{side}$ ,  $d_{det}$ ,  $d_{excl}$ ,  $d_{st}$ ,  $w_r$ ,  $d_{rel}$ ,  $d_{min}$  and  $d_{max}$ . These values basically depend on the dimensions of the car and the nature of the task (character of obstacles). Note that listed parameters influence only the performance of the collision avoidance system, as the ability to reach the preset destination depends solely on extended TMU (demonstrated in section III.A) which is currently optimized for given car dimensions.

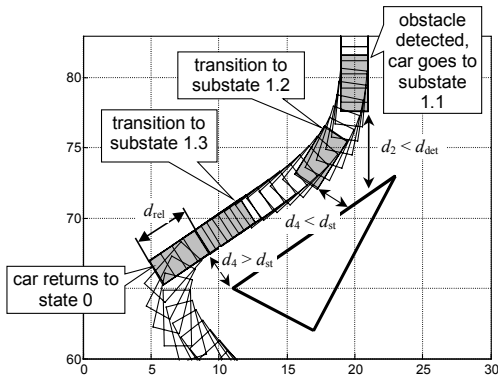


Fig. 12. State transition in obstacle avoidance.

In parameter settings it is possible to detect internal dependencies.  $w_r$  must be chosen such that the car would leave reasonable distance between itself and the obstacle wall (something like few car widths) and in present setting this value is chosen to be 4 meters. The value of the detection limit  $d_{det}$  partially derives from the former i.e. it must be large enough (in current setting this parameter equals 6 meters) to provide smooth trajectory from the detection point to the point where the car establishes  $w_r$  (Fig. 13, right). On the other hand, it is important to keep this parameter as small as possible not to make the car to react to the obstacles at longer distances (Fig. 13, left). Obviously,  $d_{st}$  (in current setting 9 meters) must be larger than  $w_r$  to make sense and similar to  $d_{det}$  (the reason is indicated in Fig. 14, left).  $d_{side}$  must also be in the same range (in current setting 8 meters) as its too large value can lead to incorrect strategical decisions (see Fig. 14, right). Release distance  $d_{rel}$  is equal to 0 in backward driving mode but 2 meters in forward driving mode because the turning radius of the car is smaller in the latter case. The remaining parameters -  $d_{min} = 5$ ,  $d_{max} = 80$ ,  $d_{excl} = 30$  - are obtained rather empirically and/or in a few test drives.

Numerically, the driving results depicted in Figs. 15-16 can be evaluated by the combined measure introduced in [9]. This measure, however, evaluates the final position of the car

that primarily depends on TMU (its ability to provide smooth and effective car trajectories is documented in [9]). The performance of the collision avoidance system what is under closer inspection here has binary character (success/failure), on the other hand and can easily be identified from Figs. 15-16.

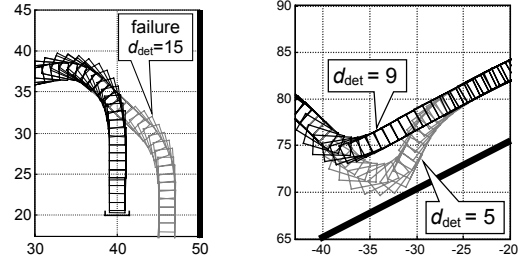


Fig. 13. Left: Too large value of  $d_{det}$  may connect the car with irrelevant obstacles (wall at right). Right: Non-smooth trajectory in substate 1.1 resulting from too small value of  $d_{det}$  (compared to  $w_r = 6$ )

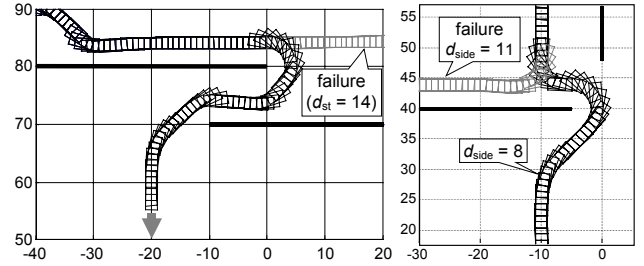


Fig. 14. Left: car does not come out of substate 1.2 if  $d_{st}$  is too large. Right: car fails to break out from the maze if the setting of  $d_{side}$  is improper.

#### V. DISCUSSION

Despite good performance in presented simulations, it is clear that there exist limits on what the proposed control system can do. First – the test assignments are based on the reasonable assumption that they must be solvable under current capacities of the control system. E.g. there must be enough space for the car to perform necessary maneuvers.

Moreover, although the designed subsystems have highly cooperative nature, there are situations where the car is unable to perform the expected task because of the conflict of interest. Fig. 17 presents one such case. At point a, the obstacle is recognized and the car starts its usual routine to avoid it. At point b, it encounters another obstacle and acts accordingly. The car is released at point c but because of the global goal (loading dock) the car performs left turn, apparently unable to find a solution to the problem.



Fig. 15. Driving backward in the field.

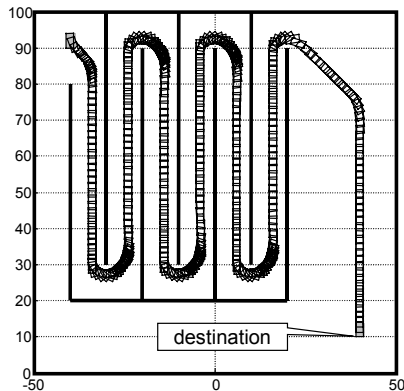


Fig. 16. Driving forward through the labyrinth.

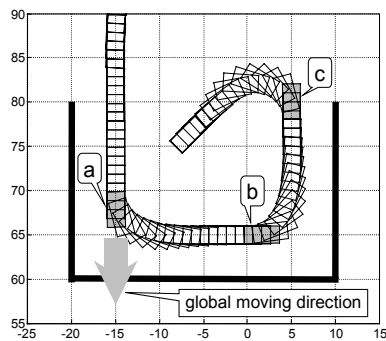


Fig. 17. Failure situation.

For all these kind of situations, another extension or “higher level of intelligence” for the car control system is needed. It should be able to recognize a failure and make more general decisions about the actions to be taken, i.e. when the car recognizes that it arrives in the position it has been before, it should force the system to solve the problem in alternative way and apply it at the right moment (point c).

Secondly, the existing system performs in the noise-free environment. In real life, however, all computations that depend on measured data are subject to noise. Presently, this affects both car position determination and decisions based on distance sensor information.

The paper is a preliminary project for car (or a similar moving object, such as mobile robot) navigation system implementation in real life and at this stage of the project we do not know what kind of noise corresponds to the conditions we are going to encounter in real life. It is possible to make the existing system less sensitive to noise if the need arises, either by sensor information processing, appropriate TMU modification or more conservative (i.e. larger) system parameter values (this actually has been tested with white noise and found to be useful for collision avoidance system).

## VI. CONCLUSIONS

In this paper, a fuzzy logic enhanced car navigation and collision avoidance system has been designed. Essentially, car control in this system is based on the flexible use of fuzzy trajectory mapping unit that enables smooth car trajectory management independent of car’s initial position or the position of the loading dock. For performing more demanding tasks, however, additional blocks of “intelligence” are required. The latter is quite possible thanks to the modular structure of the control system that also enables us to tune subsystems of the control system responsible for different tasks in separate without jeopardizing overall performance.

## REFERENCES

- [1] D. Nguyen, and B. Widrow, "The truck backer-upper: An example of self-learning in neural network," *IEEE Contr. Syst. Mag.*, vol. 10, no. 2, pp. 18-23, 1990.
- [2] S.-G. Kong, and B. Kosko, "Adaptive Fuzzy Systems for Backing up a Truck-and-Trailer," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, pp. 211-223, 1992.
- [3] L.-X. Wang, and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. on System, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414-1427, 1992.
- [4] P.A. Ramamoorthy, and S. Huang, "Fuzzy Expert Systems vs. Neural Networks – Truck Backer-Upper Control Revisited," in *Proc. IEEE Int. Conf. on Systems Engineering*, pp. 221-224, 1991.
- [5] J.J. Shann, and H.C. Fu, "A fuzzy neural network for rule acquiring on fuzzy control systems," *Fuzzy Sets and Systems*, vol. 71, pp. 345-357, 1995.
- [6] A. Ismail, and E.A.G. Abu-Khousa, "A Comparative Study of Fuzzy Logic and Neural Network Control of the Truck Backer-Upper System," in *Proc. IEEE Int. Symp. On Intelligent Control*, 1996, pp. 520-523.
- [7] I. Dumitrache, and B. Catalin, "Genetic learning of fuzzy controllers," *Mathematics and Computers in Simulation*, vol. 49, pp. 13-26, 1999.
- [8] M.-C. Su, and H.-T. Chang, "Application of neural networks incorporated with real-valued genetic algorithms in knowledge acquisition," *Fuzzy Sets and Systems*, vol. 112, pp. 85-97, 2000.
- [9] A. Riid, and E. Rüstern, "Fuzzy logic in control: truck backer-upper problem revisited," in *Proc. 10th IEEE Int. Conf. on Fuzzy Systems*, 2001, vol. 1, pp. 513-516.
- [10] A. Riid, and E. Rüstern, "Fuzzy Hierarchical Control of Truck and Trailer," in *Proc. 8th Biennial Baltic Electr. Conf.*, 2002, pp. 141-144.