Intelligent Autonomous Driving

Popa Alexandra, Toderean Bianca, Toderean Liana-Maria, Ulici Ioana-Anamaria, Rusu-Both Roxana, Miclea Liviu-Cristian
Automation Department
Technical University of Cluj-Napoca, Cluj-Napoca, Romania
(roxana.both@aut.utcluj.ro, ulici.ioana@yahoo.com, bianca.toderean@gmail.com)

Abstract— **Machine learning is an emerging technology that can be used in multiple domains, including autonomous driving. This approach is explored in the present study case of 1:10 scale car components, delivering performances and further thoughts on full-scale implementation of an artificial neural network in the area of self-driving automobiles. The frames taken from live recording pass through an artificial neural network and return a steering angle. Even though the components used had limited specifications, the artificial neural network proved successful, with quite high accuracy in keeping the car on track, parking and maintaining smooth drive.**

Keywords— artificial neural networks, autonomous driving, machine learning

I. Introduction

Commonly used methods for autonomous driving assistance systems (ADAS) count on Global Navigation Satellite System (GNSS) or Differential Global Positioning Systems, but knowledge derived from it is not sufficiently detailed and precise. Therefore a fusion of inertial systems with GNSS was used for many years in the autonomous driving development [1]. Smartphones' capabilities have also been tried [2], but the accuracy and independency offered by these approaches are below the necessities. Perception of the car improved when sensors were added to the vehicle's structure and thus new lines of action and research emerged, such as visual odometry and Simultaneous Localization and Mapping (SLAM), [3].

Improvement in the field is still undergoing, with advances in 3D laser scanning [4] for road boundaries and lane line detection or 2D laser images used in continuous measurement and evaluation of lane-based pavement distress [5]. These kinds of approaches avoid as much background noise as possible, making it possible to work even off-road, or eliminating the iterative trial-and-error parameters setting and computations. However, a long period of time spent and working on techniques such as using Digital Highway Data Vehicle (DHDV), new thresholding strategies of Linear Support vector Machine (LVSM) [5] need to compensate for the good performances. Even methods of Model Predictive Control are used for autonomous driving in different conditions, [6].

Full-scale autonomous driving is addressed by an increased number of companies, as it may be the technology decreasing the number of automobile deaths. The main approach used in the present time is sensor fusion, information inputted from lidar, radar, ultrasound and cameras being processed into a decisional process that ensures safe navigation. However, Tesla and Google have used in public statements ideas and keywords regarding artificial intelligence based on artificial neural networks [7, 8]. The synergy of the sensors cumulates into a tested basis that can be built upon. Therefore, machine learning is gaining momentum in this field. The chosen onset for the task of designing a self-driving 1:10 car was also artificial neural networks.

A short commentary on artificial neural networks will be further presented. The main idea is to simulate densely interconnected brain cells to get the computer learn by itself [9]. The constructive unit of a neural network is the neuron that computes simple calculations and sends the information to interconnected neurons. They are arranged in layers that connect only to neighboring layers, each fulfilling a specific task with a respective function of the weights attached to it. The information enters each layer and proceeds to the next one in a processed manner, until an outcome is issued. Each network contains an input and output layer and all the other hidden layers. The input units receive forms of data that the network will attempt to learn about and process, while the output units signal how it responds to learnt information, [9]. Training a neural network basically means calibrating all of the weights by repeating two steps, forward propagation and back propagation. For forward propagation, a set of weights is applied to the input data and an output is calculated. Weights can be selected randomly, as null, or as an intuitive approximation. For back propagation, the margin of error of the output is measured and weights are adjusted to decrease the error. Thus, these methods are repeated until the weights are calibrated accurately to predict an output, [10]. Activation functions in each layer decide if the neurons are fired or not. This activation is needed because the neuron does not know the ranges in which our network should work, so a function is imposed to make outer connections consider the neuron activated. An ideal function would be one that determines sparse and correct activations, therefore creating a more efficient and lighter network.

In 2017 MIT published a research in which the consumer interest in self-driving cars was presented. 48% of the 3000 participants of the study said they would never purchase a car that completely drives itself. The data suggest a proportional shift away from comfort with full automation, [11]. That may be understood as this is a technology that deals with people's lives. However, it is a technology that may be improved until it delivers positive results over time and in various tests.

This paper takes into account the limitations of the small and moderately powerful components, both hardware and software, and applies the technology of artificial neural networks, folded over the specifications. The approach is used to investigate a more plain and entry level of control and processing, from which to further build upon.

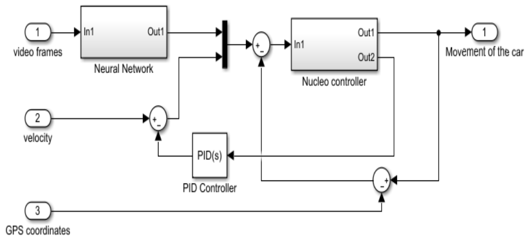


Fig. 1. Block Diagram of the process

In Fig.1 a MATLAB Simulink Block Diagram of the whole process is represented which outlines the functional blocks for a better understanding. The inputs from the camera, the frames, are used by the neural network to output a predicted angle. The Nucleo controller is used to process the velocity, the GPS coordinated and the predicted angle measurements.

In the following sections, functionality and results will be elaborated. In the System Specifications section, details about the components, the logic behind, the testing expectations and procedure will be given. The Results and discussions section provides observations on the car behavior and further improvements. Finally some concluding remarks are presented in the Conclusions section.

II. SYSTEM SPECIFICATIONS

A. *Components*

The 1:10 scale car is composed of energy supplied in the form of LiPo battery, a ReelyElektroOnroad-Chassis ARR with wheels, a model vehicle body, an RC 540 Reely electric motor, a steering servo drive, STM Nucleo Controller board, Raspberry Pi board, electric motor driver, incremental rotary encoder. Fig.2 presents the components' layout. The 2 controllers are fixed in the back of the car, over the nested electric motor that powers the back wheels. The encoder is put in a relatively more open space, and the battery is secured in the other side. The forward-inner section is more open to allow the camera to be placed in its support and for its cable strap to not be interfered with. The servo motor controlling the front wheels is located underneath the frontal chassis part.

Regarding dimensions, the car has 20cm width, 43cm length and 15cm height, while the considered road is 73cm wide, with 2 lanes of 35cm each and white line marks of 1cm width on margins and in the middle.

B. *Navigation*

The navigation is based on the developed artificial neural network, using the Raspberry Pi camera V2 available, and no other sensors for the time being. Specifically, staying in own lane with a smooth and continuous drive, without abrupt and too frequent steering was accomplished by training an artificial neural network of 9 layers developed using Python 3.2 environment.

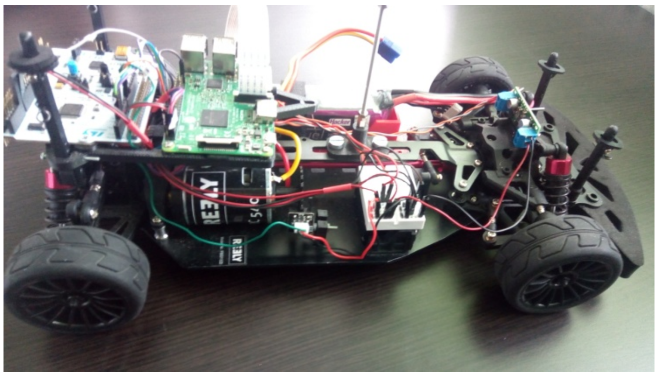


Fig. 2. Car components layout

This study case used a supervised learning regression training that enters the domain of behavioral cloning, a technique that takes decision inputs from a process and simulates the decisions in similar environments.

The road the car drives onto contains white lines as boundaries and middle delimitation of the 2 directions, all over a dark surface, similar to real life conditions. Constraints regarding memory and processing frequency of the Raspberry Pi processor lead to such a small architecture. However, the dimensions were enough for driving us to relevant conclusions. Using available artificial neural networks architecture several trials were made, but the dimensions of the neural networks exceeded those of a favorable outcome. The architecture was influenced by NVIDIA article "End to End Learning for Self-Driving Cars", [12]. Actual training of the network was made on an Intel I7 processor of a separate machine, having taken into consideration Raspberry Pi's specifications. This was made possible since the training module outputs a model file that is transmittable. The file, with extension ".ckpt" also enables a continuous use of the network, being able to further improve it by training over the already existent model. Further expanding on the architecture of the convolutional neural network (CNN), represented in Fig. 1, the final version of the developed neural network consists of inputs as the pixels from the live camera frames, a normalization layer, 5 convolutional layers, 3 fully connected layers and outputs that return the steering angle for the captured situation. The normalization layer maps the angles in [-1, 1] range. The convolutional layers answer to the purpose of feature extraction. The first 3 use kernels of dimensions 5x5x3 and a stride of 2, extracting the plain features, while the last 2 convolutional layers have kernels of size 3x3 with no stride, choosing out the more sophisticated features. After the last convolutional layer the activation map is flattened to match the input type of the fully connected layers, from multidimensional array to 1-dimensional array.

For training, a dropout of 15% was used, disabled during testing.

The loss function computes the mean-square between the prediction and the labels and an Adam optimizer finds the most influencing weights on the output and updates them to the corrected values. The activation function used is the classic nonlinear ReLu, as it handles most of the situations well, with its function:

$$A(x)= \max(0, x) \quad [1]$$

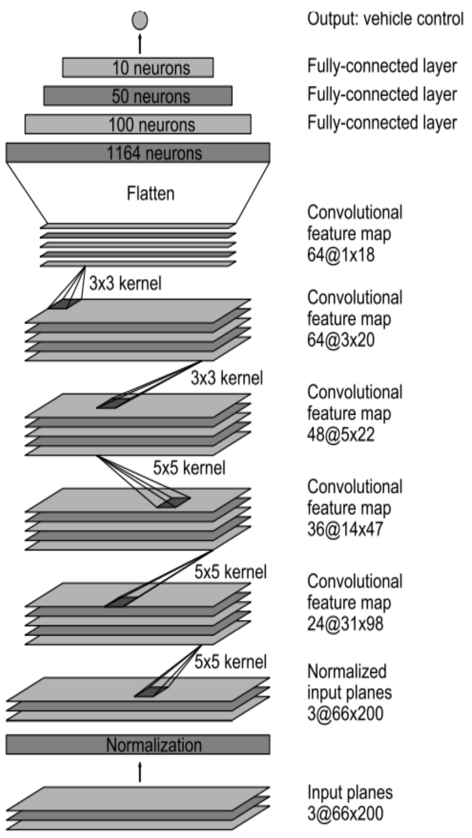


Fig. 3. Convolutional neural network architecture. The network has about 27 million connections and 250 thousand parameters [6]

The sequential training underwent a number of 30 epochs on ~6000 photos. An epoch is a measure of number of times the weights are updated in the training. The number of the epochs and the learning rate of 1e-4 were chosen after sustaining several tests and assigning a value that secured the absence of underfitting and overfitting. To remove a bias towards driving straight, the training data includes a higher proportion of frames that represent road curves [12]. The photos were taken using two tools to drive the scaled car on the road: a gamepad application and a keyboard application. Each of the frames captured needed to have an angle assigned to, and values from joystick and respectively keyboard controls were saved for this usage. However, several inconsistencies arose: the dependency on the battery level of the number of frames relative to the number of angles, the different frame rate of the camera to the frequency with which the values were cached, the angle in a range too small to work with accuracy. Therefore much of the volume of assignment process, and thus database, was inserted manually and human error may have infiltrated the project. On the other hand, manual modifications were needed, since if erroneous angles are saved from the automatic assignment with the joystick and keyboard, they will go through the loss function during training and update the weights accordingly, deteriorating the prediction. To avoid as much as possible environmental agents of distraction, data preprocessing was required. The format needed was BGR, as the camera records in this color space, and a zone of interest was cropped to only train on smaller images that comprised the area over the car's front body.

After all the preprocessing and after passing through the artificial neural network, the resulted program achieves 4 fps.

C. GPS-like system

A video camera based "GPS like" navigation feedback system was installed over the track in order to provide geo-spatial positioning. It allowed the vehicle to determine its location and rotation as complex numbers in relation to the track as a reference system, with a granularity of 1 second. Thus, a parallel with a normal satellite system was implemented and brought in possibilities of developing an autonomous drive closer to the conventional standards.

D. Decisional process

While the camera warms up, the car determines the ideal path to the points given as destination. The map of the track is available, Fig.4, just as in real-life situations, but takes the form of a json file which contains the nodes of an oriented graph. Each node of the graph contains 7 fields: name, coordinates, the connections with the other nodes (ahead, back, left, right) and the type of the node (intersection or lane). Based on this oriented graph, an adjacency list was made. Each arch from the adjacency list has the weight 1, except the intersections which have increased weight to create the possibility of finding a path with fewer intersections, using Dijkstra algorithm. The current node and the previous node are known at each step. The algorithm assumes adding the visited nodes in a queue, marking the *visited* field that certifies their presence in it. When a deadlock is reached, the current node is eliminated from the queue and another unvisited possible node is found. At the end of the execution of the algorithm, the queue contains the list of the nodes leading to the destination point. It also knows when an intersection will occur on its path (the nodes from the intersections are marked in the graph). Afterwards, the car calculates the angle between the current point, the next point and the point after the intersection. If the angle is zero, it will go ahead. Otherwise, it needs to steer. Efficient steering requires a correction regarding the point from which it will start the action. The artificial neural network is not used for steering in the intersection. Instead, Bezier curves of 4 points taken from experimentation were used to take on the desired trajectory. However, since they are not variable, an ideal point of starting to steer is needed, thus the correction mentioned has been implemented, also taking into consideration the granularity of 1 second of the GPS. The car preventively stops in any intersection, but stations more if it recognizes a stop sign. For that matter, template matching in OpenCV was used, with a threshold of 0,7. A template image, a stop sign photo, is searched in the source image, each frame taken by the camera. The template is slid pixel by pixel through the image source. At each location a correlation metric is calculated and stored in a result matrix which is further looked into to find the highest value. The higher these values are, the bigger the possibility to find the template in the respective areas. Calibration can be done by choosing different thresholds to compare the result matrix values to, and determine the presence of the template. After successfully changing directions, the algorithm returns to the artificial neural network.

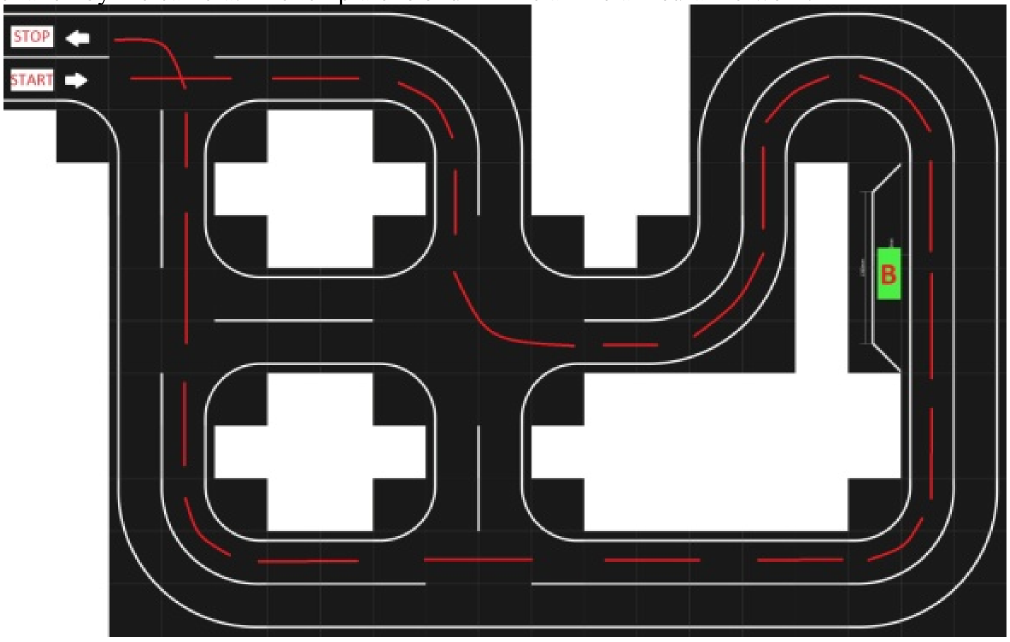


Fig. 4. Racetrack example with dimensions 8m/5.5m and coordinates from GPS

Another task for the car is parking. The coordinates of the center of the parking lot were known, so a function dependent on both the current coordinates and the ideal point from which to park could be made. From that point, 2 Bezier curves were applied for parallel parking, specifically a maximum curve to the right (22 degrees being the constructive limit of the wheels' rotation) and next a maximum curve to the left (-22 degrees), like a driver would park with its back laterally. The recognition of the park sign is decisive, since it would not park if there is technically no permission. Throughout the route, the car drives with a speed of 25cm/s, regulated by a PID controller.

E. Testing

For this matter, physical racetracks were used, as in Fig.4. The START and STOP locations are placed on the figure, as well as the B point where the parking is to be performed. These locations are given beforehand, as a driver knows where he starts his journey and where he wants to arrive. The abrupt red line indicates the trajectory calculated by the car as best and also taken by the vehicle, passing through the big intersection only once, since it has a higher weight as the smaller intersections. From point to point there are some numbers indicating the coordinates the GPS-like system gives when in that position. Fig.5 shows the racetrack from a spectator's perspective, not as a schema. The first tests comprised of neural networks trained on only about 300 photos and 15 epochs, but it was clear that the approach may work, since, even though the trainings were that small, the car had only few derailments, that were resolved by modifying the database. Another take on testing was coding a pseudo-GPS, mimicking the one implemented on the official racetrack, with keyboard-inputted coordinates.

Before this, the algorithm for path finding was also tested on the computer, and further included in the manual GPS algorithm. After several successful tests, the codes were included in the main program of the car.

Fig. 6 and 7 present a scenario of testing, with Fig.3 having encircled the predicted angle for the frame in the car's perspective from Fig.4. The minus sign states the direction, left in this case. More specific, when the frame represented is taken from the camera and passed through the neural network, the car decides that it should take a left turn with a 15 degrees angle.

The car has not been tested with obstacles on the road, since the algorithms for such a case have not yet been implemented.

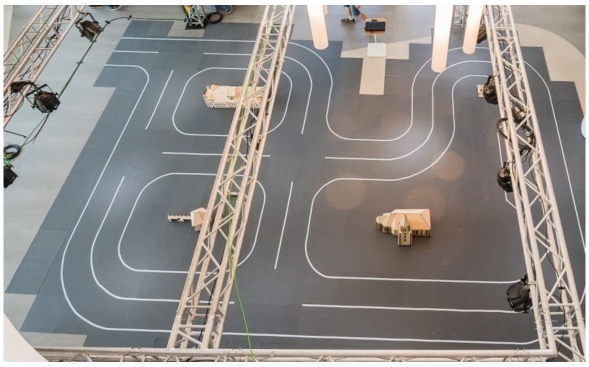
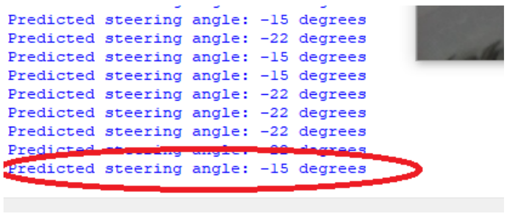


Fig. 5. Racetrack aerial view



```
Predicted steering angle: -15 degrees
Predicted steering angle: -22 degrees
Predicted steering angle: -15 degrees
Predicted steering angle: -15 degrees
Predicted steering angle: -22 degrees
Predicted steering angle: -22 degrees
Predicted steering angle: -22 degrees
Predicted steering angle: -22 degrees
Predicted steering angle: -15 degrees
```

Fig. 6. Predicted angle

Fig. 7. Car's perspective

III. Results and Discussions

The lane tracking was successful, but only under certain conditions. Since the training images were taken only once, at a certain time of the day, the different lighting affected the performances. The network had the option to have the outputs calibrated, but only to a certain point.

What the network really needs is significantly more datasets, with different environmental conditions, from lighting, to obstacles, etc. A more powerful and faster mean of data processing would also be useful. To that effect, Movidius stick and restriction of python libraries would be mentioned.

The Movidius stick is a GPU (Graphics Processing Unit) that can be directly attached to the Raspberry for better graphical processing. Using only Numpy and Scipy Python libraries could provide a much faster result.

In our case, the libraries included in the project count OpenCV and TensorFlow, but employing only mathematical logic with the mentioned Python libraries would lead to a far better use of the memory and time. Another optimization would be variable Bezier curves and variable parking. In the study case, these were fixed, and the results were dependent on luck too, specifically depending on where the car would stop and what orientation it would have, since even a 2% change in these parameters would generate a different outcome. However, it is valuable that the car also managed to maintain itself in its lane on a different arrangement of the track. Another result worth mentioning is the first place in Bosch Future Mobility Challenge 2018, a competition specific in automating this type of car.

In Fig.8 is presented how the car behaved when in new environment lighting, which, in this particular case, was distant to data-sampling lighting. From a 3 minute drive, the time for which the car followed its lane was gathered and the 2 were compared.

It was shown that in the beginning, until the calibration was fully done, the car had some derailments, since different turns appeared in its path, but from a certain moment, in which we declared the calibration was finished, the car had no trouble following the lines.

Fig.8 recreates a path without any intersections, focusing only on the way the light influences the car and how the car behaves during calibration. Fig.9 focuses on the car going on different paths that do contain intersections, but the data was taken in the absence of the GPS feedback. With this setup, the graph shows how the cars ability to pass through intersections. The majority of the iterations involve a number of successful passings fewer than the total passings.

This effect was caused by the presence of the big intersection in the path of the car. Even if the car often manages to successfully pass by an intersection, there are times when the outcome is unfavorable. This issue is to be solved with the GPS-like system that concerns itself with the ideal positioning of the car when an intersection follows.

The generalization of the project is possible within a certain condition, one of them being a varied database. In this case, as we mentioned, it was necessary to calibrate the neural network if the testing environment had different lighting than the data gathering environment. However, the order and length of curves and the length of the straight road had no influence, as it proved to behave similar with different arrangements. The lighting was what influenced the abilities of the car. Another requirement would be for the car to start on the track, as if it is positioned off-road, it moves, at a first glance, randomly. It does not move literally random, since it actually tries to find anything that would resemble its known environment of white lines over a darker surface. It has been observed that, if on the track rays of sunlight unveiled parallel to the white borders, in the middle of the road, the car would act as if the light was the actual border and try to keep itself between them.

IV. Conclusions

The final result of the project is a 1:10 scale car that is able to keep itself on designated lane, with curves, for an unspecified amount of time, conditioned that the lines are continuous and no interrupt of the path is present. It has also spawned a result of often but not always being able to deal with intersections, conditioned by a GPS-like system with high accuracy that can communicate with the car. These results have come to be due to the artificial neural network implemented on the car.

It is worth emphasizing that the scale of the project is reduced. However, the components have also restrictions with respect to these dimensions. A full scale car that also owns components, hardware and software with much better performances and additional body parts is likely to manage similar full-scale situations. Further research on such projects deserves to be made, since, with the limited components this case study had, it delivered satisfying performances. A much broader database would be needed, incorporating also different situations, not only lane keeping.

The approach of neural networks dives into the environment layers of the car deeper and more detailed, with better reliability than classical vision processing and may build up the path to a real autonomous car.

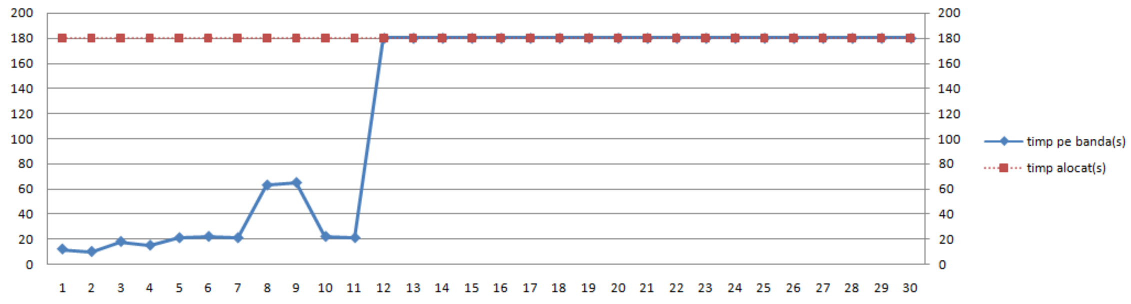


Fig. 8. Time spent on track (blue) from a total predefined time (red), in this case 3 minutes

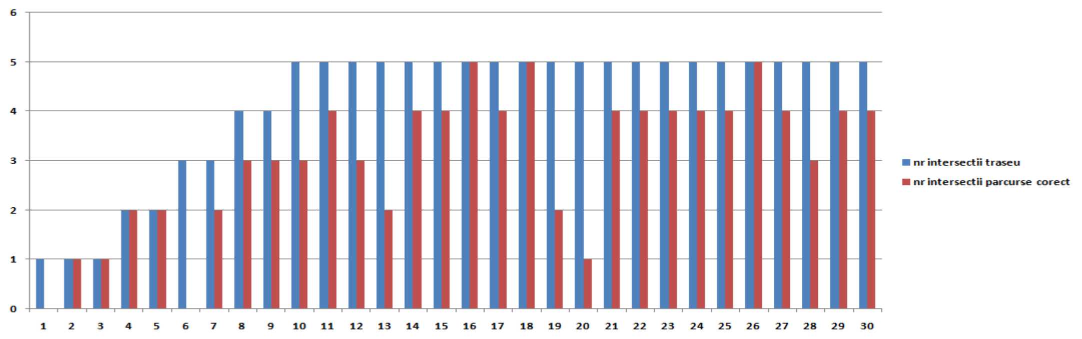


Fig. 9. Number of intersections correctly passed (red)from total number of intersections from test run (blue)

ACKNOWLEDGMENTS

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS – UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0734, contract number 72PCCDI⁄2018.

Expressing thanks also to the Bosch Engineering Center Cluj.

REFERENCES

[1] J.Z. Sasiadek, P.Hartana, “GPS/INS Sensor Fusion for Accurate Positioning and Navigation Based on Kalman Filtering”, IFAC Proc. Vol. 37, 2004

[2] P.A.Zandbergen, “Accuracy of iPhone locations: A comparison of assisted GPS, WiFi and cellular positioning” Trans. GIS, vol.13, 2009

[3] F.demim, A. Nemra, K. Louadj, “Robust SVSF-SLAM for Unmanned Vehicle in Unknown Environment”, IFAC-PapersOnLine , vol. 49, 2016

[4] F. Jiemenez, M. Clavijo, F. Castellanos, C. Avarez, “Accurate and Detailed Transversal Road Section Characteristics Extraction Using Lase Scanner”, Appl. Sci. 2018, 8(5), 724

[5] L. Li, W. Luo, K.C.P. Wang, “Lane Marking Detection and Reconstruction wity Line-Scan Imaging Data”, Sensors (Basel). 2018 May 20;18(5)

[6] K.Liu, J. Gong, S. Chen, Y. Zhang, H. Chen, “Model Predictive Stabilization Control of High-Speed Autonomous Ground Vehicles Considering the Effect of Road Topography”, Appl. Sci. 2018, 8(5)

[7] Tesla autopilot homepage, www.tesla.com/en_EU/AUTOPILOT

[8] Google waymo tech webpage, https://waymo.com/

[9] C. Woodford, “Neural Networks”, www.explainthatstuff.com/introduction-to-neural-networks.html

[10] S. Miller, “Mind: How to build a Neural Network (Part One)” https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/

[11] H. Abraham, B. Reimer, B. Seppelt, C. Fitzgerald, B. Mehler& J. F. Coughlin, “Consumer Interest in Automation: Change over One Year”, Transportation Research Board 97th Annual Meeting, Project: Trust and Vehicle Technology, 2018

[12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K Zieba, “End to End Learning for Self-Driving Cars”, Cornell University Library, arXiv:1604.07316