Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 175 (2020) 453–458

Procedia Computer Science

www.elsevier.com/locate/procedia

The 15th International Conference on Future Networks and Communications (FNC)
August 9-12, 2020, Leuven, Belgium

Design and simulation of vehicle controllers through genetic algorithms

Amelec Viloria [a]*, Nelson Alberto Lizardo Zelaya[b], Noel Varela[c]

[a,c] *Universidad de la Costa, Barranquilla, Colombia.*
[b] *Universidad Tecnológica Centroamericana (UNITEC), San Pedro Sula, Honduras*

Abstract

Genetic Programming (GP) is a population-based evolutionary technique, which, unlike a Genetic Algorithm (GA) does not work on a fixed-length data structure, but on a variable-length structure and aims to evolve functions, models or programs, rather than finding a set of parameters. There are different histories of driver development, so different proposals of the use of PG to evolve driver structures are presented. In the case of an autonomous vehicle, the development of a steering controller is complex in the sense that it is a non-linear system, and the control actions are very limited by the maximum angle allowed by the steering wheels. This paper presents the development of an autonomous vehicle controller with Ackermann steering evolved by means of Genetic Programming.

Keywords: Design and simulation; Vehicle controllers; Genetic algorithms.

1. Introduction

The research aims at generating the steering controller of a stand-alone vehicle. It is proposed work on a straight-line path generated by a previous information processing for reducing the difficulty of handling relative coordinates, as used in [1]. On the other hand, it is contrasted with this study in which curves are used and the set of trajectories

* Corresponding author. Tel.: +57-3046238313.
E-mail address: aviloria7@cuc.edu.co

1877-0509
10.1016/j.procs.2020.07.064

with which it is trained is reduced. From this point of view, the proposed controller is a bit complex, since the system has a desired trajectory as input and at the controller output there is only an interaction with the angle of the front wheels. Most probably, if the PG is given to solve the whole problem, it would not be able to solve it or it would require too much computing power. So, it was divided into two parts [2].

The first part is to achieve a controller capable of making the vehicle follow a given straight line, from any position and orientation relative to it. However, this sub-problem can be further simplified. If it is considered that the car must follow the X axis, it can be generalized to any line by doing the corresponding coordinate transformation. Also, the controller does not need to be run for any position and orientation, since the path will be kept close to the car all the time [4]. This way, only positions within 2m of the line and orientations of maximum 90° and minimum -90° were used [5]. The second sub-problem would then be a controller deciding when to switch from following a line to following the next one. The paths are generated randomly for each generation, which promotes the generalization of the result.

For the simulation in both cases, the time had to be discretized, and it was done with intervals of a tenth of a second of resolution [6]. In the same way, the resolution is enough to make a faithful simulation. Since the speed of the car was considered at all times 1m/s, there is a displacement of only 10cm for each evaluation [7][8].

2. PG parameters

For the first sub-problem, an evaluation of the controller was taken in each interval and the value given was considered as the speed with which the direction of the wheels was changed in radians/second. The simulation was run for 20 minutes, after which 100 generations were executed. The evaluation function was scalar multi-objective [9], in which 3 parameters were taken into account: the integral of the curve described by the car (divided by 100); the difference between the oscillations made and the minimum necessary (divided by 15); and the final error of the car [10].

For the second one, two different versions were made. In both, if the result of the individual's evaluation is positive, it is decided to change to the next line, otherwise, the car stays on the current line. The calculations were made with the coordinates relative to the line in which the car determined to be. For the first version, the simulation took a little over an hour to carry out 100 generations [11].

The evaluation function again was scalar multi-objective, taking into account the difference between the oscillations made and the minimum necessary (divided by 5); the integral of the curve described by the car (divided by 10); and, considering that the trajectory was given in an arrangement of straight lines, the difference between the index of the straight line it reached and the one it should have reached, multiplied by 100. This high weight was used to force the simulation to take the individual to the end of the trajectory. To obtain a better controller, the individual was evaluated on 10 different initial positions and was assigned the worst of the 10 as an adaptation score [12].

For the second version, a driver capable of avoiding collisions was already being sought. It was considered to add sensors as variables or sensing forms as functions. However, the need for processing increased too much. So, it was considered at the end to use trajectory boundary lines placed at 3 meters from the trajectory. The goal now would be to stay within those lines, where going outside would be considered a collision. Thus, the path generator prior to this controller should only make sure that the obstacles are outside these lines [13].

In the simulation, only 50 generations were carried out in about 3 hours. This is because the calculation of collisions took a lot of computing resources. The objective function was defined only as the minimization of collisions. In this case, each individual was evaluated on 10 different trajectories, and the individual's score was taken as the average. Unlike the two previous simulations, care was taken to ensure that, although the trajectories were different in each generation, they were the same for each individual, thus making a fairer classification that would lead to faster convergence [14].

The parameters shared by all the simulations are shown below, together with the base functions (Table 1), and the variables taken into account in the simulation (Table 2) [15]:

- Population size: 500
- Crossover probability: 0.8
- Mutation probability: 0.1

- Number of generations: 100
- Maximum depth:
  - For first sub-problem: 7
  - For second version 1: Adf0: 2; Adf1: 2; Main shaft: 4
  - For second version 2: Afd0: 3; Adf1: 3: Main tree: 7
- Elitism: no
- Selection form: Tournament size 2

Table 1. Functions used.

Function	Result
Negative(x)	-x
Sum (x, y)	x+y
Subtraction (x, y)	x-y
Mul (x, y)	x*y
Division (x, y)	x/y yes $y \neq 0$ 1if y=0
IfR (a, b, c, d, e)	d yes $b \leq a \leq c$ and otherwise
IfGT (a, b, c, d)	c yes $a \geq b$ d otherwise
Hyp (x, y)	$\sqrt{x^2 + y^2}$
Sin(x)	sin(x)
E(x)	$e\text{-}abs(x)$
Root2(x)	$\sqrt{x}$

3. Analysis of results

The first controller was crucial for the continuation of the investigation, as the final result would be based on it. Different simulations were run with different configurations; the final one is the one described in the previous section. In addition to adjusting those parameters, 10 different simulations were run with them; and from these, the one that gave the best results was chosen, based directly on the error and the appearance of the trajectory, shown in Figure 1:

Neg(Substract(Mul(E(THr),Division(Y,L)),Subtract(Neg(THr),IfR(X,Ms,THa,W,THa)).

The response of the system with this controller can be seen and very little oscillation is observed when trying to follow the Xn axis.

In this controller, some variables whose presence could be easily predicted can be observed [16]. For example, the Y-coordinate tells the control how far away it is from the line; the angular velocity helps to avoid overshooting. On the other hand, it is interesting to see the presence of the variable X, since, indirectly it expresses how the error is different at the beginning of the path. This is because, within a conditional function, this variable modifies the result only in a range of values.

For the second sub-problem, since 2 different versions were created, it was possible to make a comparison of what is generated with PG by changing the target function [7]. Even though both versions solve the same thing, the way their performance is rated is different, therefore, the definition of "best" changes for each simulation, which can be seen in Figures 2 and 3.

Table 2. Variables used in the simulation.

Variables	Representation
Length of the car	L
Width of the car	A
Speed	V
Wheel angle	THr
Distance in the direction of the straight line	X
Distance to the line	Y
Orientation of the car	THa
Maximum wheel torque	Ms
Speed in the direction of the straight line	Vx
Speed perpendicular to the line	Vy
Car Angular Speed	W
Distance to the next corner	Desq
Length of next line	D_Sesq
Angle of the next corner	Thesq
Next angle to Thesq	Th_Sesq

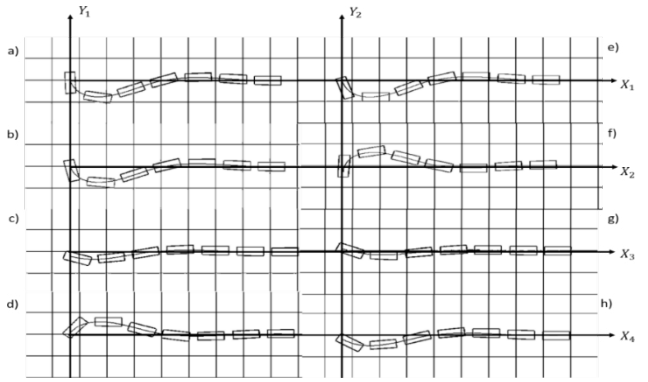

Fig. 1. Three different simulations with different initial conditions are shown (in 5x5m grid).

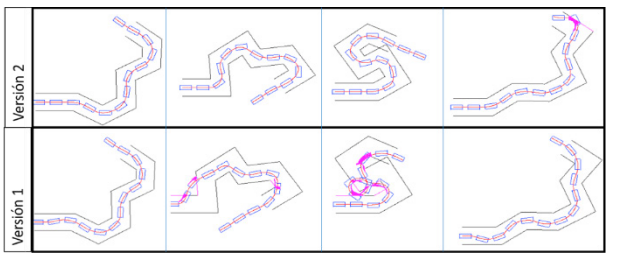

Fig. 2. Comparison of the two versions of the controllers.

It can be seen how version one, by being mainly guided by the integral of the curve, evolves a strategy that seeks to find shortcuts. Version two, in contrast, by being guided by the collisions, remains more attached to the trajectory it must follow.

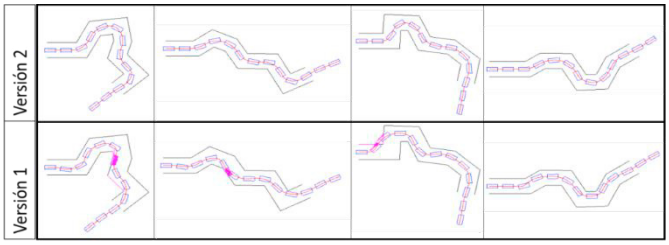

Fig. 3. Comparison of the two versions of the controllers.

A similar behavior to that presented in Figure 2 can be observed. Thus, the different controllers obtained were the following:

- Version 1:
  - ADF0:
    - Sum(A,A)
  - ADF1:
    - Hyp(Desq,THa)
  - Main tree:
    - IfR(Th_Sesq,ADF1),Neg(D_Sesq),Sine(ADF0,Desq)), Root2(V),IfR(D_Sesq,ADF0,ADF0,THa,-0.659634))
- Version 2:
  - ADF0:
    - IfR(Division(X,V),IfR(V,-1.422355,V,1.950899,V),IfR(Thesq,X,X,-1.422355,X),-1.422355,-1.422355)
  - ADF1:
    - Hyp(IfR(Ms,Vx,D_Sesq,ADF0,D_Sesq),Ms)
  - Main Tree:
    - Subtract(Subtract(Sum(Desq,Mv)),Sum(Desq,ADF0)),Root2(Neg(Hyp(Desq , Hyp(Sum(Desq,ADF0)),Sum(THr,ADF0))))

The different strategies followed in versions 1 and 2 have advantages and disadvantages. On the right side of Figure 2 it can be seen how the shortcut preference of version 1 gives it an advantage over version 2. However, in the rest of the paths, the same preference generates a disadvantage, especially in the central images; while in the left image a path with more oscillations is still generated.

4. Conclusions and future research

Genetic programming was successfully used to develop a controller, which was tested under different circumstances and proved to solve the problem. This is regardless of the configuration of the trajectory, and to some extent, its complexity, since the controller was able to stay within the boundary lines in most of cases. The applicability of genetic programming in controller design can be observed. In this case, it was started with a quite big search space, using all the variables that could or could not be involved; and the same was done with the functions. And even with such a vast search space, the algorithm was able to deliver an acceptable result. It should also be noted that the separation of a problem into sub-problems is important. In this way, even though there is a large space for solutions, the problems that are being attempted to be solved are relatively simple. For example, at the beginning, it was considered to make a controller that would take the car from point A to point B. However, that involved too many variables, and it was a

very complex problem so no results were obtained. Thus, the need to divide the problem into two parts was noted. Another point to take into account is the limitation of computer resources. For version 2 of the second sub-problem, the use of sensor information was considered, but this made the need for computing resources too great, to the point that each generation took more than 20 minutes to evaluate. Thus, while the controller was not able to use environmental information directly, it was able to avoid obstacles indirectly.

References

[1] Kasparavičiūtė, G., Nielsen, S. A., Boruah, D., Nordin, P., & Dancu, A. (2018, July). Plastic Grabber: Underwater Autonomous Vehicle Simulation for Plastic Objects Retrieval Using Genetic Programming. In International Conference on Business Information Systems (pp. 527-533). Springer, Cham.

[2] Li, R., Noack, B. R., Cordier, L., Borée, J., Kaiser, E., & Harambat, F. (2017). Linear genetic programming control for strongly nonlinear dynamics with frequency crosstalk. arXiv preprint arXiv:1705.00367.

[3] Li, R. (2017). Aerodynamic Drag Reduction of a Square-Back Car Model Using Linear Genetic Programming and Physic-Based Control (Doctoral dissertation).

[4] Li, R., Noack, B. R., Cordier, L., Borée, J., & Harambat, F. (2017). Drag reduction of a car model by linear genetic programming control. Experiments in Fluids, 58(8), 103.

[5] Hein, D., Udluft, S., & Runkler, T. A. (2018). Interpretable policies for reinforcement learning by genetic programming. Engineering Applications of Artificial Intelligence, 76, 158-169.

[6] Bartczuk, Ł., Łapa, K., & Koprinkova-Hristova, P. (2016, June). A new method for generating of fuzzy rules for the nonlinear modelling based on semantic genetic programming. In International Conference on Artificial Intelligence and Soft Computing (pp. 262-278). Springer, Cham.

[7] Yusuf, R., Podusenko, A., Tanev, I., & Shimohara, K. (2018, November). Recognition of mistaken pedal pressing based on pedal pressing behavior by using genetic programming. In 2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS) (pp. 104-108). IEEE.

[8] Ji, X., He, X., Lv, C., Liu, Y., & Wu, J. (2018). Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits. Control Engineering Practice, 76, 41-53.

[9] Phan, D., Bab-Hadiashar, A., Lai, C. Y., Crawford, B., Hoseinnezhad, R., Jazar, R. N., & Khayyam, H. (2020). Intelligent energy management system for conventional autonomous vehicles. Energy, 191, 116476.

[10] Lam, A. Y., Leung, Y. W., & Chu, X. (2016). Autonomous-vehicle public transportation system: scheduling and admission control. IEEE Transactions on Intelligent Transportation Systems, 17(5), 1210-1226.

[11] Alekseeva, N., Tanev, I., & Shimohara, K. (2019, July). On the Emergence of Oscillations in the Evolved Autosteering of a Car on Slippery Roads. In 2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) (pp. 1371-1378). IEEE.

[12] Vásquez C. et al. (2020) Conglomerates of Bus Rapid Transit in Latin American Countries. In: Pandian A., Ntalianis K., Palanisamy R. (eds) Intelligent Computing, Information and Control Systems. ICICCS 2019. Advances in Intelligent Systems and Computing, vol 1039. Springer, Cham

[13] van Lon, R. R., Branke, J., & Holvoet, T. (2018). Optimizing agents with genetic programming: an evaluation of hyper-heuristics in dynamic real-time logistics. Genetic programming and evolvable machines, 19(1-2), 93-120.

[14] Boslough, M. (2017, March). Autonomous dynamic soaring. In 2017 IEEE Aerospace Conference (pp. 1-20). IEEE.

[15] Mrugala, K., Tuptuk, N., & Hailes, S. (2017). Evolving attackers against wireless sensor networks using genetic programming. IET Wireless Sensor Systems, 7(4), 113-122.

[16] Viloria A. et al. (2019) Analyzing and Predicting Power Consumption Profiles Using Big Data. In: Wang G., Bhuiyan M., De Capitani di Vimercati S., Ren Y. (eds) Dependability in Sensor, Cloud, and Big Data Systems and Applications. DependSys 2019. Communications in Computer and Information Science, vol 1123. Springer, Singapore.