A Traction Control System based on Co-evolutionary Learning in Spiking Neural Network (SNN)

Conference Paper · July 2018

3 authors:

Javier Pérez Fernández
University of Malaga
12 PUBLICATIONS 55 CITATIONS

Juan Cabrera
University of Malaga
69 PUBLICATIONS 940 CITATIONS

Juan Castillo
University of Malaga
36 PUBLICATIONS 423 CITATIONS

A Traction Control System based on Co-evolutionary Learning in Spiking Neural Network (SNN)

Javier Pérez, Juan A. Cabrera & Juan J. Castillo ´
Department of Mechanical Engineering
University of Malaga, Malaga, Spain
E-mail : javierperez@uma.es

A traction control system (TCS) is designed and trained for different road conditions with co-evolutionary learning based on genetic algorithms. Common TCS do not consider the variation and oscillation created in the transition between surfaces, defining a control logic which is highly dependent on road accuracy and a speed estimations. To solve this problem, a co-evolutionary learning process is used. The proposed procedure trains the control algorithm, which is based on a spiking neural network, on different surfaces and surface transitions looking for the worst-case scenario. Therefore, a control algorithm with a good dynamic response to constant and changing roads is obtained. This control makes the system stable when the road estimation is delayed or unstable, solving a common flaw produced by sensor noise or computation delays.

Keywords: Co-evolutionary, Traction Control System, Spiking Neural Network

Powertrain Control and Energy Management

1. INTRODUCTION

Active safety systems in vehicles are an important tool to reduce accidents, by either controlling the slip of the wheels or increasing stability. Traction control systems plays an important role during accelerations, controlling the slip and maintaining the wheel-asphalt contact to improve vehicle handling. Changes in the type of road may cause an unwanted response due to transitory oscillations. Usually control algorithms perform better when the road adherence is constant. However, oscillations are observed when there are transitions between different surfaces. This event can cause an accident due to loss of traction or reduce vehicle performance. In motorcycles, TCS are less common due to the complexity of motorcycle dynamics and the increase in cost. In general, TCS control signals are rear tyre torque, input throttle and/or wheel/motor rpm. These parameters have to be measured or estimated in combustion motorcycles. However, in electric motorcycles, they are easily obtained with a high degree of accuracy. Therefore, TCS is basically only a software problem in these vehicles. In this work, an algorithm capable of controlling the slip for electric motorcycles is described. Among the different techniques used in TCS algorithms are Slip Threshold, PID, Sliding Control, Fuzzy Logic and Neural Network. The slip threshold uses a simple algorithm, [1], where a maximum slip ratio is established depending on the road type that enables the TCS to cut off the power to the wheels. The main problem of this control is the dependence on road estimation, making the control even worse in transitions due to lack of accuracy and delays of the estimator. PID controllers, [2], also depend heavily on estimation but perform better thanks to the time dependence of the derivative and integrator parameters.
Sliding Control, [3], [4], tracks the control variable while varying through time, improving dynamic response with the same estimation dependence as the PID. Fuzzy Logic introduces the experience of an expert [5]. The control algorithm also requires more information about the current state of the vehicle such as throttle position or torque in the drive wheels. Therefore, the estimator variables have less importance in this logic.

The fusion between the PID and the fuzzy controller increases the capabilities of the control offering better response, but the logic is still not adequate to manage road shifts [6]. Furthermore, neural network control, [7], is capable of handling variation in road conditions due to their plasticity that adapts the control during transitions. Neurofuzzy combination is also used to define the control algorithm, [8], [9].

Three problems appear when using neural networks-based controllers. First, time dependence, classical artificial neural network needs a few delayed inputs to respond to quick variation. Second, the neural structure in neural networks defines the control and learning capabilities. Third, the learning process affects the ability to adapt to different roads. The training set teaches the control how to perform on a road, being difficult to obtain a good behavior on various surfaces. After analyzing these problems, our implementation is presented. Spiking neural networks are introduced as a reliable model of real neurons found in nature. The use of these neurons in control is quite extended, translating control behavior found in humans or other animals to engineering control issues [10], such as self-driving [11], robotics [12] or biomechanics [13]. Other works focus on vehicle dynamics, [14], without considering transient conditions. This paper presents a spiking neural network control with co-evolutionary learning that improves the response of the system when sudden road changes occur. SNNs make use of temporal spike trains to command inputs and outputs, allowing a faster and more complex computation. Co-evolutionary learning is used to train the network in different situations through the worst-case scenario. After an iterative process, the obtained network makes the control stable in transitions.

The rest of this paper is organized as follows. In Section 2 the motorcycle model used is defined. In Section 3, the traction control system is presented. Section 4 describes a Control Spiking Neural Network structure while Section 5 gives the details on the co-evolutionary learning process. In Section 6, experimental results are presented. Conclusions and future works are discussed in Section 7.

2. MOTORCYCLE MODEL

An electric motorcycle is modelled in ©BikeSim and ©Matlab. The model uses a torque-rpm map to reproduce the performance of the electric motor in traction and regenerative braking processes.

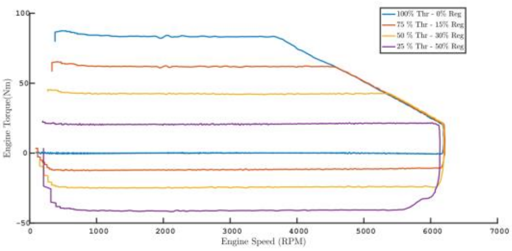


Fig. 1. Motor torque-rpm map for throttle and regeneration values

3. TRACTION CONTROL SYSTEM OPERATION

The operation of a TCS focuses on maintaining the optimal wheel slip for each road type. This assures the maximum road friction coefficient. Rear wheel slip is obtained from equation 1.

$$s_r = 1 - \frac{V_x}{\omega_r R_r} \tag{1}$$

Where, ω_r is the rear tyre speed, R_r is the wheel radius and V_x is the vehicle speed. Vehicle speed is considered to be known in this work. Previous research show that GPS, EKF or a combination of them can be used to estimate the speed. In this paper, vehicle speed data provided by BikeSim is used.

Once the rear slip has been obtained, the control algorithm only needs the optimal slip to set a target. Optimal slip calculation introduces another level of complexity due to road type dependence. A fuzzy estimator developed by our research group in [5] is used to obtain the optimal slip.

4. CONTROL SPIKING NEURAL NETWORK

This section focuses on the control algorithm structure based on spiking neural networks. The neuron, synapse model and the arrangement of input and output neurons are defined.

a. Neuron Model

An Izhikevich neuron is used. This model accurately reproduces a real neuron apart from its low computation requirements. Two differential equations imitate membrane potential 2 and the recovery value of neuron 3. The threshold where the neuron is fired, is set by equation 4 resetting both membrane potential v and recovery u.

$$\frac{dv}{dt} = 0.94v^2 + 5v + 140 - u + I(t) \tag{2}$$

$$\frac{du}{dt} = a(bv - u) \tag{3}$$

$$\text{If } v \geq 30 \ \ then \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{4}$$

With this set of equations, the neuron type is defined by parameters *a, b, c, d* as Izhikevich defines, creating different kinds of neuron behavior depending on the input current *I(t)* from the previous synapses.

b. Synapse Model

The synapse model determines how the neural network learns and evolves into an optimal control logic. For this reason, the model should accurately represent a synapse in real nature. Neuron interconnections perform a complex process but could be simplified as follows.

$$I(t) = w_{ij}\varepsilon(t - t_{spike}) \tag{5}$$

$$\varepsilon(t - t_{spike}) \begin{cases} \dfrac{t - t_{spike}}{\tau} e^{1 - \frac{t - t_{spike}}{\tau}}, & if \ \ t \geq t_{spike} \\ 0, & if \, t < t_{spike} \end{cases} \tag{6}$$

Where I(t) is the output currents that flow to the post synaptic neurons as a result of the input spike timing (t_{spike}) and the weight of synapse w_{ij} that control the strength of the connection. As in nature, the output current decreases by time *t* in a exponential decay controlled by τ .

c. Neural Network Structure

Neural structure depends on its application. One or more hidden layers are used when a net is designed to detect a pattern or to recognize a kind of behavior. This way, the performance of the network is increased, but also its complexity. However, in control applications the net should be as simple as possible to respond to input variations quickly. For this reason, the proposed net (Fig. 2) only has two layers: an input layer with three neurons and an output layer with two neurons.

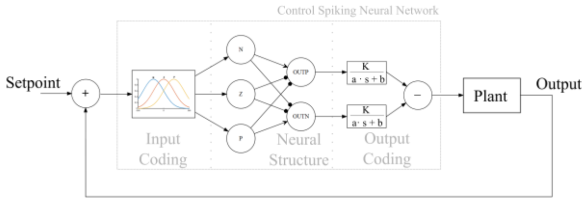


Fig. 2 Neural Network Structure

Because spiking neurons work with spikes, a coding and decoding stage is needed. The lowest possible delay should be added by this process. An input code similar to a fuzzification stage with gaussian bells is used. In addition, an output code similar to two antagonistic muscles model is used [10]. This way, the algorithm tries to imitate natural control structures, where a few neurons respond quickly to a given input. The desired behavior maintains the output while the input error is zero, increasing the output when the input is low and vice-versa. The weight in the synapse connections is capable of exciting or inhibiting the post-synaptic neuron depending on the sign. In Fig. 2 inhibited connections are represented by circles and excite connections by arrows.

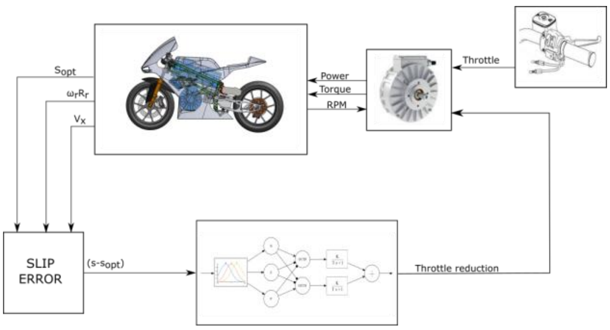


Fig. 3 Control Neural Network Scheme

The whole scheme (Fig. 3) is simulated in ©Simulink and BikeSim. Different roads and learning methods are tested. Results are shown in section 5 and 6.

5. CO-EVOLUTIONARY LEARNING

The learning process improves control performance by training the system in different conditions. The main goal is obtaining a control algorithm capable of performing optimally on different roads and road changes during acceleration. The ratio between theoretical distance and obtained distance is used to perform comparisons.

a. Target Function

The longitudinal equation of the motorcycle is used to obtain the theoretical distance. Aerodynamic force F_{aero} and rolling resistance force F_{roll} are not considered.

$$F_{fric} - \cancel{F_{aero}} - \cancel{F_{roll}} = m\,a_1 = \mu_1 F_z = \mu_1 k_r mg \qquad (7)$$

After replacing longitudinal friction force F_{fric} by wheel friction equation $F_{fric} = \mu F_z$, where μ is the friction coefficient and Fz is the vertical load on the drive wheel. A fraction k_r of the total mass m is taken as the vertical load on the rear wheel. Thus, maximum possible acceleration can be calculated from equation 8 is obtained and. The constant acceleration motion equation yields the theoretical maximum distance 9.

$$a_1 = \mu_1 g k_r \qquad (8)$$

$$mt_1 = V_{x_0}\Delta t_1 + \frac{1}{2}a_1(\Delta t_1)^2 \qquad (9)$$

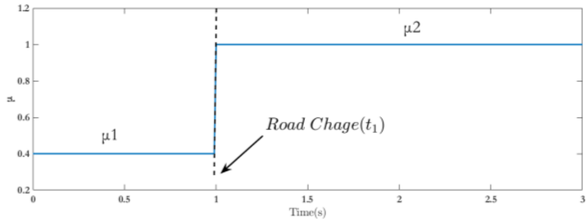


Fig. 4. Test definition

A variable test is introduced to get a stable control while the road changes. Fig.4 shows reproduces this fact. Friction coefficient μ_1 is used for the first section of the road and friction coefficient μ_2 for the second section of the road. For this test, total error 10 is the quadratic error of the two sections of the road 11 and 12:

$$e(\mu_{1,}t_{1,}\mu_{2,}) = \left(\left(\frac{m_1}{mt_1}\right)^2 + \left(\frac{m_2}{mt_2}\right)^2\right)^2 \qquad (10)$$

$$mt_1 = V_{1x_0}(t_1 - t_{init}) + \frac{1}{2}\mu_1 g k_r (t_1 - t_{init})^2 \qquad (11)$$

$$mt_2 = V_{2x_0}(t_{end} - t_1) + \frac{1}{2}\mu_2 g k_r (t_{end} - t_1)^2 \qquad (12)$$

Where m_1 and m_2 are the traveled distance for each controller, and mt_1 and mt_2 are the theoretical maximum distance. The learning procedure tries is designed to minimize this error.

b. Learning Procedure

Co-evolutionary learning (Fig. 5) is implemented as the learning algorithm to reduce the error in a supervised learning process. A given set of inputs and outputs are used to train the net and another set is used to validate the learning process. In control application, the neural algorithm is trained in a close loop, where the inputs are roads or road variations and the output is the target function. The TCS is aimed to reduce the slip error which produces a longer travelled distance.

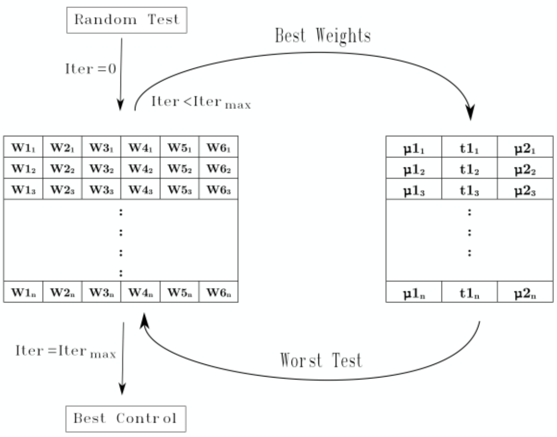


Fig. 5 Learning Process

In co-evolution, two or more species reciprocally affect each other's evolution. In this paper, co-evolution Genetic algorithms are used to evolve the species.

Our neural network is the main species. The other one is road condition based on the target function. Both species compete in a prey and predator interaction. The prey is the neural net that evolves for a given predator (road condition), until it becomes invincible, with an optimal result for this road combination. Then the predator for the given prey (optimal algorithm) evolves and defeats it, finding the road condition where it performs the worst, thus defeating the prey.

This evolution procedure is repeated iteratively. Our proposed structure has 3 input neurons and 2 output neurons, so a total of 6 weights need to be set. The other parameters are the neuron type that is fixed during simulation. The road condition as presented in Fig. 4 is defined by 3 parameters (μ_1, t_1, μ_2). All these parameters are selected by the algorithm to get the best results, as will be shown in the next section.

6. RESULTS

Finally, to demonstrate the capabilities of our control, different road tests are simulated (Table 1 and 2). The co-evolutionary based learning process is compared with a genetic algorithm trained for a specific road. A genetic algorithm trained for different road conditions is not used due to the much higher computational effort needed to get a similar error. A fuzzy logic controller is also used to compare the control with another approach.

Two main simulations are used to evaluate the capabilities of the neural network controller. First a constant road condition test is shown in Fig. 7. Then a variable road condition simulation is done in Fig. 8. On one hand, the results for the constant road condition are obvious. The neural network controllers trained with genetic algorithms perform better on the roads where they were trained. However, their performance is worse with other road conditions. Fuzzy logic yields good results on all roads. Finally, the algorithm trained with co-evolution achieves the better overall result.

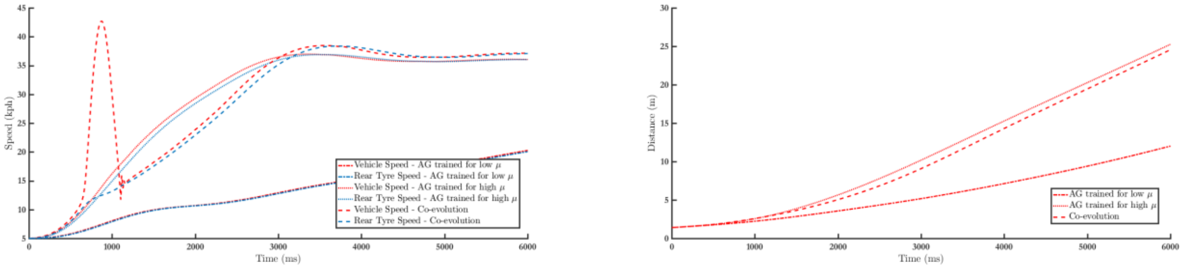
Table 1 Error value for constant road test

Road	**μ = 0.2**	**μ = 0.4**	**μ = 0.6**	**μ = 0.8**	**μ = 1**	**Σe**
AG Train μ = 0.2	**1.0000**	0.8379	0.8267	0.7171	0.6237	4.0055
AG Train μ = 0.4	0.9793	**1.0000**	0.8719	0.8685	0.7360	4.4556
AG Train μ = 0.6	0.9183	0.8268	**1.0000**	0.7875	0.6276	4.1601
AG Train μ = 0.8	0.9666	0.8619	0.9220	**1.0000**	0.8407	4.5911
AG Train μ = 1	0.9017	0.7667	0.9090	0.8884	**1.0000**	4.4658
Co-evolutionary	0.9540	0.8851	0.9428	0.8639	0.9705	**4.6162**
Fuzzy Logic	0.9260	0.8314	0.9203	0.7885	0.6417	4.1079

Table 2 Error value for variable road test (t_1=1.5s)

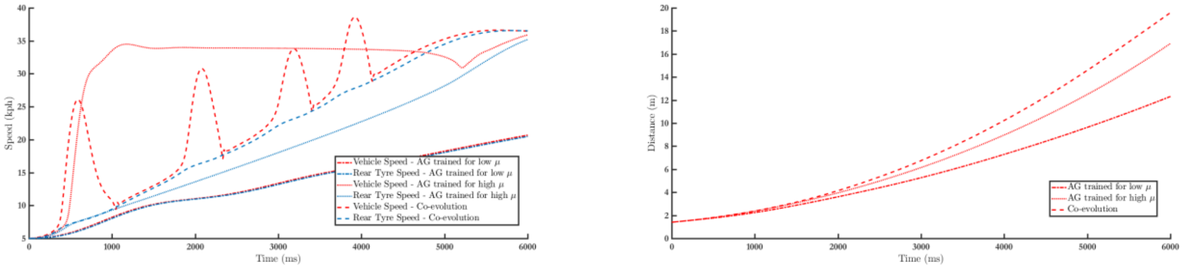
Road	**μ = 0.2 μ = 0.6**	**μ = 0.6 μ = 0.2**	**μ = 1 μ = 0.4**	**μ = 0.4 μ = 1**	**μ = 0.6 μ = 0.4**	**μ = 0.4 μ = 0.6**	**Σe**
AG Train μ = 0.2	0.8177	0.7705	0.6337	0.9391	0.8092	0.8966	4.8668
AG Train μ = 0.4	**1.0000**	0.8776	0.7468	0.8847	0.8808	**1.0000**	5.3898
AG Train μ = 0.6	0.9250	**1.0000**	0.6721	0.9677	**1.0000**	0.9504	5.5152
AG Train μ = 0.8	0.8591	0.9112	0.8415	0.8480	0.9053	0.9328	5.2979
AG Train μ = 1	0.8533	0.8449	**1.0000**	**1.0000**	0.8554	0.8628	5.4164
Co-evolutionary	0.9292	0.9359	0.9689	0.9084	0.9481	0.9444	**5.6349**
Fuzzy Logic	0.9440	0.9965	0.6840	0.7431	0.9727	0.8737	5.2141

a. Constant Road



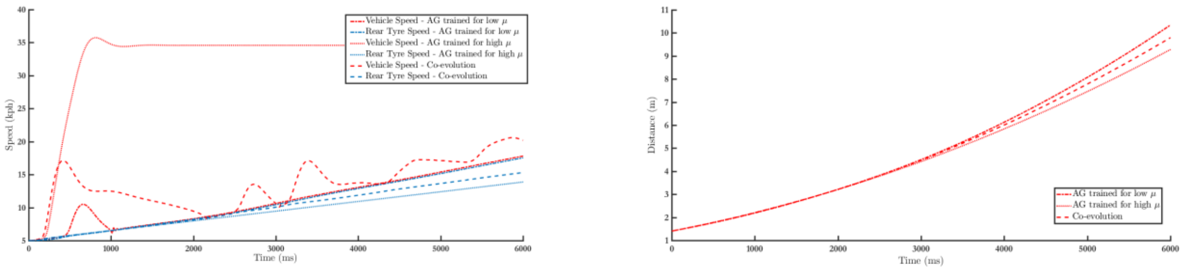
a) b)

Fig. 7.1 Constant road test with high adhesion (μ_1=μ_2=0.8)



a) b)

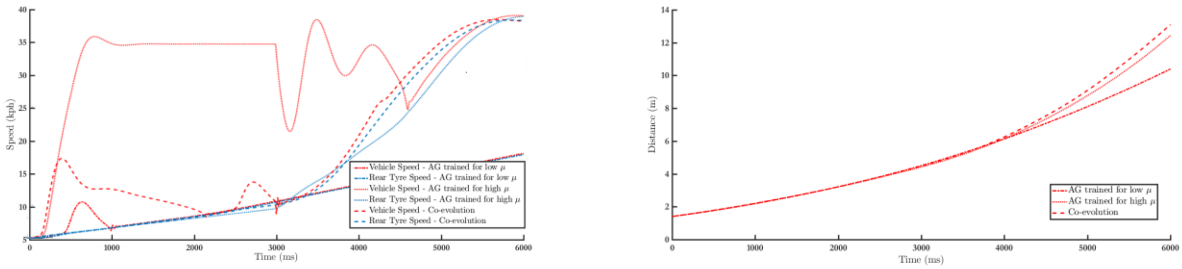
Fig. 7.2 Constant road test with medium adhesion (μ_1=μ_2=0.6)



a) b)

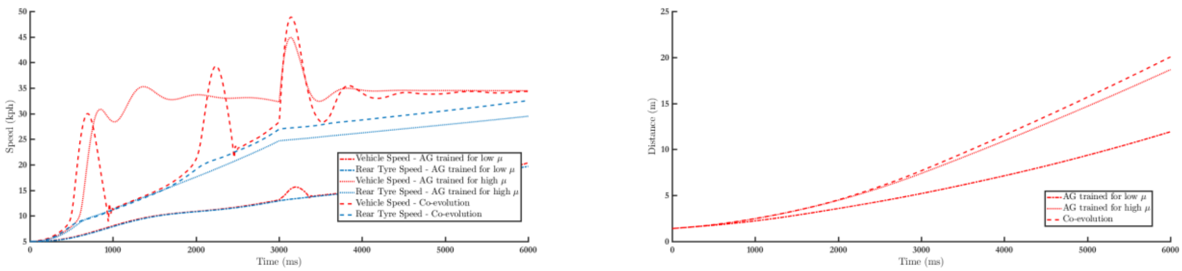
Fig. 7.3 Constant road test with low adhesion (μ_1=μ_2=0.4)

b. Variable Road



a) b)

Fig. 8.1 Variable road test with adhesion values from low to high (μ_1=0.4 →μ_2=0.8)



a) b)

Fig. 8.1 Variable road test from high to low adhesion (μ_1=0.8 →μ_2=0.4)

* a) Vehicle and wheel speed for low and high adhesion controllers trained with AG in comparison with the co-evolutionary controller. b) Distance in meters for each acceleration.

Similar conclusions can be drawn when road changes take place in the middle of the simulation. In this case, the GA-trained controllers performance is highly dependent on the initial surface. Again, Fuzzy Logic provides good results in all cases. Finally, co-evolution has the best overall performance.

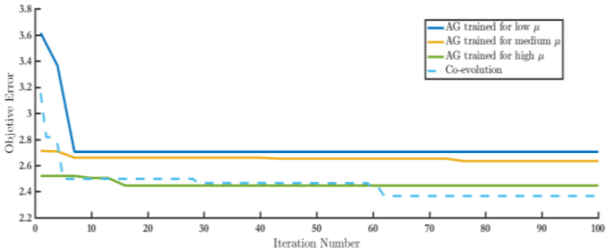


Fig. 6. Learning Process

After a few iterations the co-evolution algorithm has the lowest for all different road conditions (Fig 6). The error of the control trained with GA for a specific road type is also shown. The performance of the training method based on co-evolution is faster than with GA. The final error using co-evolution is lower for the same computation effort and execution time.

7. CONCLUSION

A Traction Control System based on Spiking Neural Networks is presented in this piece of work. Genetic and co-evolutionary algorithms are used to train the neural network. The learning process based on co-evolution reduces the computational time of the training process. Conventional TCS do not behave optimally in all road conditions and, specially, when road adherence changes during its activation. The proposed controller has the best overall performance not only in constant adherence conditions but also when adherence changes during the test.

In future works, tests will be carried out on real roads with an electric bike.

ACKNOWLEDGMENT

This work was supported in part by the Spanish Innovation Science Ministry under Grant TRA2015-67920-R.

REFERENCES

[1] Y. Yifan, Z. Jian, Z. Yang, and W.Jian, “Research and Test on Traction Control System of Distributed Driving Electric Vehicles,” no. 2, pp. 277–280, 2017.

[2] M. Vaezi, E. S. J. Hesam, and S. Anwar, “IMC-PID traction control system for an automobile via engine torque control,” Proceedings of the IEEE International Conference on Industrial Technology, vol. 2015-June, no. June, pp. 296– 302, 2015.

[3] Y. Tang, X. Zhang, D. Zhang, G. Zhao, and X. Guan, “Fractional order sliding mode controller design for antilock braking systems,” Neurocomputing, vol. 111, pp. 122–130, 2013.

[4] S. Rajendran, S. Spurgeon, G. Tsampardoukas, and R. Hampson, “Time-Varying Sliding Mode Control for ABS Control of an Electric Car,” IFAC-PapersOnLine, vol. 50, no. 1, pp. 8820–8825, 2017.

[5] J. A. Cabrera, J. J. Castillo, E. Carabias, and A. Ortiz, “Evolutionary optimization of a motorcycle traction control system based on fuzzy logic,” IEEE Transactions on Fuzzy Systems, vol. 23, no. 5, pp. 1594–1607, 2015.

[6] H. Dahmani, O. Pag, and A. E. Hajjaji, “Traction Control System Using a Fuzzy Representation of the Vehicle Model,” Control, no. 1, pp. 5712–5717, 2015.

[7] H. Wang, C. Xue, W. A. Gruver, and B. C. Vsa, “Neural Network Control of a Parallel,” pp. 2934–2938, 1995.

[8] Y.-e. Mao, Y. Zhang, and T. Liu, “The sliding mode controller for automotive ABS based on the fuzzy neural network,” 2010 Sixth International Conference on Natural Computation, no. Icnc, pp. 1410–1413, 2010.

[9] A. V. Topalov, Y. Oniz, E. Kayacan, and O. Kaynak, “Neuro-fuzzy control of antilock braking system using sliding mode incremental learning algorithm,” Neurocomputing, vol. 74, no. 11, pp. 1883–1893, 2011.

[10] J. Pérez, J. A. Cabrera, J. J. Castillo, and J. M. Velasco, “Bio-inspired spiking neural network for nonlinear systems control,” Neural Networks, vol. 104, pp. 15 – 25, 2018

[11] J. Kaiser, J. C. V. Tieck, C. Hubschneider, P. Wolf, M. Weber, M. Hoff, A. Friedrich, K. Wojtasik, A. Roennau, and R. Kohlhaas, “Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks,” pp. 127–134, 2016.

[12] T. S. Clawson, S. Member, S. Ferrari, S. Member, S. B. Fuller, R. J. Wood, and S. Member, “Spiking Neural Network (SNN) Control of a Flapping Insect-scale Robot,” no. Cdc, pp. 3381–3388, 2016.

[13] M. Hulea and C. F. Caruntu, “Spiking neural network for controlling the artificial muscles of a humanoid robotic arm,” 2014 18th International Conference on System Theory, Control and Computing, ICSTCC 2014, pp. 163–168, 2014.

[14] Y. Oniz, A. C. Aras, and O. Kaynak, “Control of antilock braking system using spiking neural networks,” IECON Proceedings (Industrial Electronics Conference), pp. 3422–3427, 2013.