Online velocity fluctuation of off-road wheeled mobile robots: A reinforcement learning approach

François Gauthier-Clerc, Ashley Hill, Jean Laneurit, Roland Lenain, Eric Lucet

Online velocity fluctuation of off-road wheeled mobile robots: A reinforcement learning approach

François Gauthier-Clerc[1], Ashley Hill[1], Jean Laneurit[2], Roland Lenain[2] and Éric Lucet[1]

***Abstract*— During the off-road path following of a wheeled mobile robot in presence of poor grip conditions, the longitudinal velocity should be limited in order to maintain safe navigation with limited tracking errors, while at the same time being high enough to minimize travel time. Thus, this paper presents a new approach of online speed fluctuation, capable of limiting the lateral error below a given threshold, while maximizing the longitudinal velocity. This is accomplished using a neural network trained with a reinforcement learning method. This speed modulation is done side-by-side with an existing model-based predictive steering control, using a state estimator and dynamic observers. Simulated and experimental results show a decrease in tracking error, while maintaining a consistent travel time when compared to a classical constant speed method and to a kinematic speed fluctuation method.**

I. INTRODUCTION

Mobile robotics attracts more and more interest in many applications such as agriculture [1]. In these applications, autonomous systems need to be adaptive to the natural environment with its unpredictable and heterogeneous properties. Path following is necessary for a large number of agricultural tasks such as seed planting, irrigation, or plant treatment [2], [3]. In these applications, accurate tracking is essential to avoid damage to vegetation. Travel time is also a criterion for maximizing robot availability and minimizing the overall cost of these solutions. Both of these requirements must be met by these tracking algorithms to ensure a successful application of robotics in agriculture.

A common approach for path tracking is to consider a constant speed tuned experimentally that is as fast as possible while ensuring a limit on the tracking error [4], [5]. This can make the system easier to control, but it has a significant impact on the optimality of the solution, both in terms of tracking accuracy and travel time. And the highly variable grip conditions in off-road context accentuates this issue, as certain portions the trajectory may require a very low speed. Therefore, in order to consider speed variations, the dynamics of the system must be taken into account, in particular for preventing wheel slip.

A planning approach can be used in order to implement an a priory online speed variation. This involves defining in advance either the trajectory and the speed for each key-point, or simply the speed on an already known static trajectory. This solution is often applied to robotic arms [6], [7] and it can be extended to mobile robots.

[1]Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France firstname.lastname@cea.fr

[2]Université Clermont Auvergne, Inrae, UR TSCF, Centre de Clermont-Ferrand, F-63178 Aubière, France firstname.lastname@inrae.fr

A travel time minimization approach applied to a simple dynamic model of the vehicle [8] is also possible. This technique can consider other constraints such as energy consumption for an electric car [9]. It guarantees an optimal travel time and a limited lateral error under the considered model. However, this technique is limited in an agricultural context as it does not take into account the evolution of the dynamics at the wheel ground interface.

Another solution of second order approximation on the acceleration constraints of a four-wheel model leads to an online planning that is able to manage the speed to guarantee a small tracking error [10]. Considering the speed management as an upper limit definition makes the problem easier to solve. Thus a simple model can be used, based on curvature and speed limit [11], [12]. A model of the wheel ground interaction can also be considered [13], [14], [15]. This estimate is based on the angular velocity which predicts a situation of under-steering or over-steering. However, all these methods are not sufficiently reactive to the online variation of parameters, in particular grip conditions.

Overall, the previously described solutions are limited to simple models that do not integrate the online evolution of wheel ground grip conditions. However, the online estimation of such slip parameters and cornering stiffnesses is essential to guarantee the stability of the system operating in highly variable environments.

Recent advances in machine learning and more specifically in reinforcement learning (RL) provide an efficient approach to derive a model from a set of input data and an appropriate reward function [16]. These methods take advantage of advances in deep learning that allow to use a neural network as a universal function estimator [17]. Contributions have applied such RL approach to a path following task [18], [19], [20], these examples showing the relevance of a learning process to solve mobile robotics problems.

The contribution proposed here is a neural network based online velocity controller. It is associated to an existing front steering Model-based Predictive Controller (MPC) for path tracking, that was previously implemented with a constant speed [14]. This task was previously addressed with a traditional approach [14], [13] which demonstrated the problem's complexity and the limits of this approach. The RL algorithms are used to solve this problem with more flexibility, as they are based on a data derived from a fully dynamic model with varying grip conditions.

The speed control solution is implemented in the following, a mobile robot dynamics simulator being used to train a neural network with reinforcement learning techniques

in order to generate a speed modulation algorithm from available inputs. This is achieved by minimizing an objective function of travel time and tracking errors.

This method is evaluated in simulated and real environments to assess its performance and limitations. For this, the methodology and models are described in section II, followed by the simulation and simulated results in section III, then continued with the experimental setup and results in section IV, and then finally concluding and discussing future works in section V.

II. METHOD

In this section, after a brief reminder of the steering control law, the problem of speed fluctuation during path tracking is discussed. First, a deterministic kinematic solution is proposed. Then, a new RL approach is investigated. To this end, an objective function is defined and the problems of actuator response time and neural network outputs are addressed.

A. MPC steering controller

The path following strategy is from previous works on the design of a front steering controller in agricultural context [21]. This MPC is based on a bicycle kinematic model in the horizontal plane, extended by taking into account the tyre slip angle. The following variables are expressed in the Frenet frame (see Fig. 1).

$$\delta_F = \left(\tan(\beta_R) + \frac{L}{\cos(\beta_R)} \left(\frac{c(s)\cos(\tilde{\theta})}{k} + \frac{A\cos^3(\tilde{\theta})}{k^2} \right) \right) + \beta_F \quad (1)$$

$$\text{with} \begin{cases} k = 1 - c(s)y \\ \tilde{\theta}_2 = \tilde{\theta} + \beta_R \\ A = -K_p\, y - K_d\, k\, \tan(\tilde{\theta}) + c(s)\, k\, \tan^2(\tilde{\theta}) \end{cases}$$

The variables used are the following:

- s, $c(s)$ the curvilinear abscissa of the robot projection on the path and the associated curvature,
- y, $\tilde{\theta}$ the lateral and angular deviations,
- β_F, β_R the front and rear slip angles,
- K_p, K_d the proportional and derivative gains,
- L the wheelbase.

Both slip angles $\beta_{F,R}$ and cornering stiffness coefficients $C_{F,R}$ are estimated online during the path following using a dynamic observer [22]. This observer formulation is based on a linear approximation of a Pacejka model [23] for the expression of lateral forces $F_{F,R} = C_{F,R}\beta_{F,R}$. This way, cornering stiffnesses reflect lateral grip properties of the encountered terrain.

As a varying speed will be used, the steering controller's K_p and K_d gains must evolve accordingly in order to maintain a constant reactivity of the controller, with respect to the speed value. Thus, a linear interpolation is performed in real-time between predefined expert gains for different values of static speeds and cornering stiffnesses.

One observation is that the steering controller used does not consider a pure dynamic model. This choice is motivated by the fact that a dynamic model involves too many parameters that are complex to estimate and very variable, especially in the agricultural context [24].

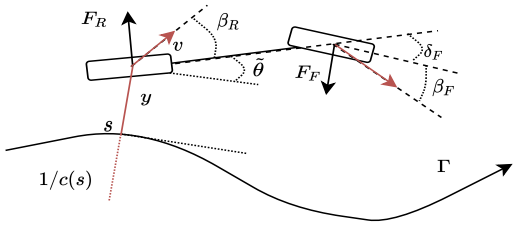

Fig. 1. Kinematic bicycle model projected on the frenet frame

B. A deterministic kinematic speed constraint solution

A deterministic method for speed modulation with a kinematic model is introduced, in order to compare to the RL approach. This technique is derived from a limitation on the steering rate $\dot{\delta}_{\text{F max}}$ (similar to [25]). The following assumptions are made: y and $\tilde{\theta}$ are assumed to be zero, which means that the robot is expected to follow its path correctly. By deriving the angular error with respect to the curvilinear abscissa, the following equations are successively deduced:

$$\frac{\partial\theta}{\partial s} = \frac{\tan(\delta_F)}{L} \quad (2)$$

$$\frac{\partial^2\theta}{\partial s^2} = \frac{1}{Lv}(1+\tan^2(\delta_F))\frac{\partial\delta_F}{\partial t} \quad (3)$$

And then, by replacing $\tan(\delta_F)$ with $L.c(s)$ (as the lateral and angular error as assumed to be zero):

$$\left|\frac{\partial\delta_F}{\partial t}\right| < \dot{\delta}_{\text{F max}} \Rightarrow \left|\frac{\partial^2\theta}{\partial s^2}\right| < \left|\frac{1+L^2c(s)^2}{Lv}\right| \dot{\delta}_{\text{F max}} \quad (4)$$

Considering that $\frac{\partial^2\theta}{\partial s^2} = \frac{\partial c(s)}{\partial s}$ and the previous equation (4), it is possible to determine a positive maximum speed depending on the desired maximum steering rate:

$$v_{max} = \frac{1+L^2c(s)^2}{L\left|\frac{\partial c(s)}{\partial s}\right|}\dot{\delta}_{\text{F max}} \quad (5)$$

As such, given a desired maximum steering rate $\dot{\delta}_{\text{F max}}$ that limits the dynamic behavior in real world experiments, a reasonable maximal speed v_{max} is obtained. However, since the rate of change of v_{max} can be greater than the maximum acceleration of the robot, a predictive speed modulation is performed to ensure that the robot speed remains below the defined v_{max}. This speed controller does not consider the estimated cornering stiffness but it does provides a relevant comparison, since the speed setpoints generated are optimal in a kinematic context.

C. Description of the selected reinforcement learning approach

The TD3 [26] reinforcement learning method is considered, in order to train a suitable speed controller based on a neural network (NN). It is an off-policy RL method that was chosen based on empirical results compared to alternative RL based methods (neural networks trained with PPO [27] and TRPO [28] obtained lower performance than TD3).

A dynamics simulator of the robot along with a sufficient diversity of trajectories, allows for off-line training of the neural network using the TD3 method. Then the trained neural network is used online, in order to generate speed

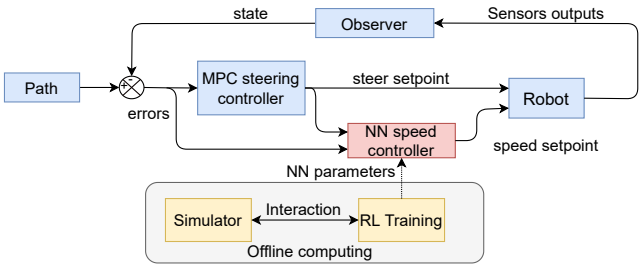

Fig. 2. The new RL approach (red and orange parts) as part of the overall path following control architecture.

setpoints and ensure a fast and accurate path following performances (see the global control architecture Fig. 2). During the online uses, there are no changes to the weights and biases of the neural network.

This velocity fluctuation management is formulated as a Markov Decision Process (MDP) with a discrete time sampling. The following neural network inputs are considered:

- y, $\tilde{\theta}$ the path tracking lateral and angular errors.
- v, $\dot{\theta}$ the measured speed and yaw rate.
- $v_{\max}$ the maximum allowed speed.
- δ_F^{odom}, $\delta_F^{\text{setpoint}}$ the measured and setpoint steer angles.
- β_F, β_R the measured front and rear slip angles.
- C_F, C_R the front and rear estimated cornering stiffnesses.
- $c(s)$ the path curvature.

The measured yaw rate is included in the input data as it is an important indicator of vehicle stability [13]. The curvature is also a crucial variable for speed modulation as it defines the admissible yaw rate and speed (see deterministic alternative section II-B). This curvature variable is extended to a vector over a fixed time horizon, allowing for the anticipation of future curvatures and estimation of its rate of change. The estimated cornering stiffness is a direct indicator of the slip dynamics at wheel ground interface. It can therefore be used by the neural network to estimate the ability of the vehicle to follow its trajectory at a given speed. Whereas the estimation of slip angles alone does not allow the method to anticipate the sliding behavior, but instead allow for the correction in case of undesirable slips.

A path tracking episode starts with a stationary robot at the beginning of a considered path. The episode ends either when the robot reaches the end of the path (successful episode), or when it reaches a state that does not respect a hard constraint (failed episode). The lateral error in this approach is defined as a hard constraint, where exceeding a maximal lateral error causes the episode to end prematurely with a low reward.

1) The reward function: RL algorithms aim to maximize a discounted cumulative reward along an episode [16]. Both traveling time and lateral error limitation must be considered in this reward function in order to correctly train the neural network model.

The reward is defined in accordance with the speed that has to be maximized, while avoiding hard constraints. The reward for the whole trajectory is the sum of the rewards at each step. In this case, a linear speed-dependent reward should be avoided since the number of steps in an episode depends directly on speed, and thus the sum of the rewards reflects the length of the trajectory, which is a desired optimization criterion. However, a square function can be used to solve this issue (see equation (6)), and to improve the convergence of the RL algorithm a constant negative reward $-C^{te}$ is added in order to encourage higher speeds.

$$R_t = \begin{cases} v^2/v_{max}^2 & \text{if } |y| < y_{\text{lim}} \\ -C^{te} & \text{else} \end{cases} \tag{6}$$

While this addition creates discontinuity, it reinforces the behavior of the policy to avoid exceeding the lateral error limit. Without this penalty, a sub-optimal policy that would rather fail at a higher speed, rather than reach the end of the path at a lower speed may emerge. The learning process on a trajectory continues until the mean episodic reward is no longer increasing.

2) Restoring the Markov hypothesis and observability of the system.: A progressive variation of the speed (limited acceleration) and the frequency of the target controller defeat the convergence of RL algorithms. Indeed, the small discretized time interval (10Hz) leads to the cancellation of the traditional RL exploration using Gaussian noise and the actuator response time affects the Markov property (causal link) [29], [30].

An action skipping is often used to address both issues and significantly improve the learning process (as it was done in previous contributions [31]) by artificially increasing the time scale during the exploration and learning process. Rather than sampling the policy at each step, the neural network evaluation is performed every k steps and kept the same action between each evaluation. This allows the exploration to maintain a consistent control setpoint (convergence of the speed towards its desired value) and to reduce the actuation delay.

In addition, the neural network can output an oscillating speed setpoint, with a potentially infinite variation. For this reason, the neural network output is interpreted as an acceleration setpoint instead of a speed setpoint. This reduces the output noise and makes it possible to add a constraint on the acceleration. Then, an integration is performed to retrieve the value of the speed setpoint. And, to preserve the observability of the system, this speed setpoint is added to the list of the following inputs.

Action skipping and acceleration output integration can both be applied in the same process by maintaining acceleration setpoint between each step instead of speed setpoint. In this way, the action skipping has no impact on the final shape of the exploration process, which is very close to an optimal policy. Fig. 3 illustrates the difference in shape between a direct speed control at 10Hz, an applied action skipping ($k = 10$), and an acceleration output consideration with the same action skipping.

III. SIMULATION

In this section, the training process using a dynamic simulator is described. Then, the constant speed, the kinematic

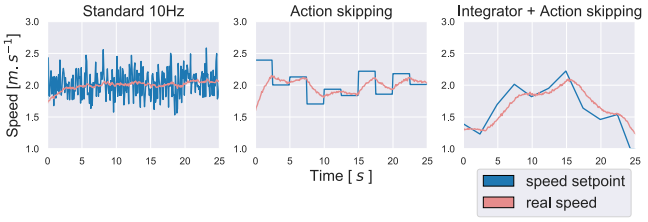

Fig. 3. A comparison of the three exploration techniques

and the NN-based methods are compared to each other, using each case the same MPC steering controller.

A. Training setup

The simulator used to calculate the TD3 algorithm is parameterized in accordance with properties of the testing robot presented in section IV. This robotic platform is a car-like robot called *Robufast* (presented in [5]). It is an electrically powered 4-wheel drive robot capable of reaching a speed of 8 m/s in off-road conditions. It can perform a maximum acceleration/deceleration of $1.5\mathrm{m\,s^{-2}}$ with a maximum steering angle of $20°$. Three maximum speed values are considered for training (4, 6, $8\mathrm{m\,s^{-1}}$). A deceleration phase is added on the top of the speed modulation to ensure a safe stop when the path end is almost reached.

The dynamic simulator integrates a linear approximation of Pacejka model with cornering stiffnesses of 7000 to $25000\mathrm{N\,rad^{-1}}$. A second-order model of actuator delay is also implemented in order to model the steering behavior.

The training path set is defined in order to represent a large set of possible paths with similar lengths, to avoid sub optimal policy, and to conform to the given specifications. In particular, the range of curvatures (0-$0.3\mathrm{m^{-1}}$) and maximum turn number (3) are fixed. Path shape and grip conditions are re-sampled using a uniform law at each episode beginning to ensure changing conditions during training, allowing for the generalization of the neural network for the task.

The training is performed under a Python application linked by a C++ hook to the simulator implemented in C++ for performance and practicality. Stable baseline [32] is used as a RL library in order to get a reliable TD3 implementation. The actor and the critic neural networks both get the same architecture, which is composed of 2 hidden layers of 64 units, an output layer of 1 units and a state dimension of 32 units. $\tanh$ is the activation function used for the whole networks. The TD3 hyperparameters used are :

- number of steps: $2.5M$
- gamma: 0.7
- action skipping: 8
- train frequency: 500
- batch size: 512
- learning rate: $4\mathrm{e}{-5}$
- buffer size: $15k$
- number of unique paths in training set: 12

B. Comparative analysis

For the speed output computation with the kinematic speed modulation method, the $\dot{\delta}_{\mathrm{F\ max}}$ value is defined empirically in order to obtain a travel time equivalent to that of the neural network based method. This tuning is done for each path.

To compare the performance of each of the methods, three metrics are introduced. The error peak represents the maximum magnitude of the lateral error during the whole path following task. The surface error outside of a corridor is defined by the criterion A_{off} in Eq. (7). And the total surface error A_{err} is defined as a special case of the A_{off} metric, with a threshold of $y_{\mathrm{lim}} = 0\mathrm{m}$.

$$A_{\mathrm{off}} = \sum_{\substack{n=0 \\ |y(t_n)|\geq y_{\mathrm{lim}}}}^{N} \left| \dot{s}(t_n) \left(\left| y(t_n) + \frac{\dot{y}(t_n)\, dt}{2} \right| - y_{\mathrm{lim}} \right) \right| dt\,[\mathrm{m^2}]$$
$$\text{with: } \begin{cases} \dot{s}(t) &= v(t)\cos(\tilde{\theta}(t)) \\ \dot{y}(t) &= v(t)\sin(\tilde{\theta}(t)) \end{cases} \tag{7}$$

A path set is used to gather an average result over multiple simulations (paths that are not used during the training process). This set is composed of 14 paths with a length of around $200\mathrm{m}$ each. Trajectories are categorized in 3 groups according to the curvature peaks. The first group A gathers all paths with curvature peaks below $0.05\mathrm{m^{-1}}$, the second one (group B) with curvature peaks from 0.05 to $0.12\mathrm{m^{-1}}$, and the last one (group C) with curvature peaks from 0.12 to $0.3\mathrm{m^{-1}}$.

Table I gathers the results obtained with the testing path set under good grip conditions. The neural network based speed controller outperforms the other methods over all the metrics. In accordance with the method design, the neural network method keeps its peak errors very close to the threshold lateral error of $0.2\mathrm{m}$, and therefore obtains an surface error above the threshold almost equal to zero.

Whereas both the static speed and the kinematic methods struggle to keep the peak errors below the $0.2\mathrm{m}$ threshold. Even though the steering MPC is designed for constant speed, the average speed is too high for proper path following during high curvature parts. This effect is even more pronounced on paths with high peaks in the curvature. The peak error increases by 170% from group A to group C.

The kinematic controller is successful in reducing the total surface error with a modulation based on the curvature. The benefit of this method increase as the peak curvature rises when compared to the static speed method. This behavior is mainly due to a lower average speed in this context, which limits the dynamic effects. Even though the kinematic controller is more efficient than the constant speed method, the neural network based controller achieves a lower surface error in the traj C category.

Fig. 4 shows the results for the category C paths using the three considered methods. The Kinematic and neural network controllers generate similar speed profiles, However two key differences are apparent. First, the neural network controller is reacting to the dynamic phenomena by decreasing the speed near $80m$ due to the lateral error increase, with the lowest speed value reached at the curvature peak. This speed reduction helps to correct the angular error induced by the curvature and keeps the lateral error below the threshold ($0.2m$). In contrast, the kinematic method does not adapt to this situation and the lateral error starts increasing near $90m$, and continues to increase until $105m$.

Secondly, the neural network controller is exploiting the characteristics of the steering MPC, as it has learned to work

Method	NN based			Kinematic speed			Static speed		
Metrics	peak error	A_{err}	A_{off}	peak error	A_{err}	A_{off}	peak error	A_{err}	A_{off}
group A	0.21±0.03	21.81±3.94	0.03±0.05	0.49±0.28	29.65±8.15	5.31±5.46	0.48±0.26	28.64±8.80	5.01±5.60
group B	0.20±0.01	20.73±2.05	0.01±0.02	0.43±0.21	27.69±6.94	4.44±3.82	0.65±0.19	31.34±9.36	8.03±6.22
group C	0.20±0.03	14.77±5.66	0.02±0.04	0.52±0.35	21.98±11.96	4.87±6.70	1.30±0.88	33.58±24.79	16.68±18.48

TABLE I

RESULTS OBTAINED ON TESTED PATHS. PEAK ERROR IS IN $[\mathrm{m}]$, A_{err} IS IN $[\mathrm{m}^2]$, AND A_{off} IS IN $[\mathrm{m}^2]$

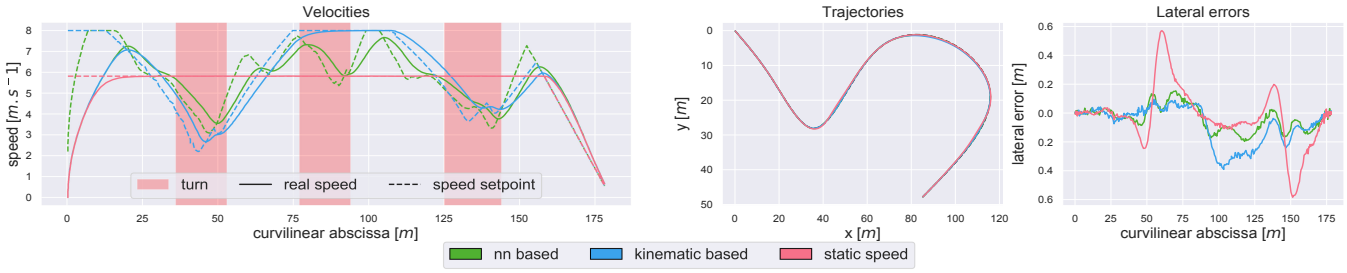

Fig. 4. Speed modulation obtained with the selected methods

Fig. 5. *Robufast* on the field

Fig. 6. trajectory shape over the field

with it during the learning process. At the beginning of a high-speed turn, the steering MPC is slightly understeering and then an oversteer to minimize lateral error. This behavior is visible around $50m$ with the static speed and the neural network controllers. Due to the fact that the kinematic controller is based on the curvature rate, it is reducing the speed value before the turn without regard to the MPC behavior, therefore the MPC cannot function optimally.

In summary, the NN speed controller has learned the subtleties of dynamics in order to avoid oversteer and understeer due to the large slip angles induced by the difficult conditions. In contrast, the kinematic speed does not take these into account, which justifies the results obtained. Further research could involve a pure dynamic MPC steering controller using an ML-based model to obtain a fairer comparison with the proposed RL speed control.

IV. EXPERIMENTAL RESULTS

Further tests were performed with the *Robufast* experimental platform (Fig. 5). Those tests aims to proove the transferability of a simulated training applied onto a real application. This comparison with and without the speed modulation, combined with the MPC steering controller, highlights the value of the proposed method.

The neural network speed controller used for this experiment was trained to avoid lateral errors above a threshold of $0.4\mathrm{m}$. It was then tested along with a constant speed strategy, over the trajectory as shown in Fig. 6 & Fig. 7. Special care was taken for these tests, in order to generate a trajectory different from those of the training process. This was done in order to verify the generalization of the neural network speed controller to different trajectories. Each test was run several times to validate the repeatability of the results, which are presented below.

Initially, the robot was positioned with a lateral error of more than one meter. The constant speed controller was set to $4\mathrm{m\,s}^{-1}$, as this achieves a similar travel time when compared to the neural network speed controller.

Results from Fig. 7 show that the neural network method is capable of maintaining the lateral error below the $0.4\mathrm{m}$ threshold, whereas the constant speed controller which has a similar travel time could not achieve a similar performance. Over the surface error metric (Eq. 7 with $y_{\text{lim}} = 0$), the neural network method has $20.35\mathrm{m}^2$ of surface error, while constant speed method has $32.98\mathrm{m}^2$ of surface error. This shows a reduction of 38.3% in the neural network method's surface error when compared to the constant speed method. As such, the neural network method was capable of significantly reducing the overall surface error and preserving the $0.4\mathrm{m}$ lateral error threshold with a similar travel time, by modulating the target speed in real time.

From Fig. 7, the neural network method seems to be decelerating the robot before each turn (red area on the velocity plot). This shows the neural network is anticipating the behavior of the robot during turning, as decelerating prevents strong dynamic effects from occurring. Furthermore, this shows that the neural network is safely adapting the speed value to the new states that were observed, which demonstrates the transferability of the method to real world experiments, as the simulation used during the training process was not a perfectly accurate robotic model.

The proposed neural network method also obtained good reactive behavior to external perturbation. At $60\mathrm{m}$ and $90\mathrm{m}$ a strong lateral error occurs due to the terrain quality in the corners, these lateral errors cause the neural network method to decelerate in order to give more time to the steering controller, allowing the robot to converge while limiting the amplitude of the lateral error.

At $100\mathrm{m}$, an oversteer occurred at the exit of the second turn, leading to large angular and lateral errors with a

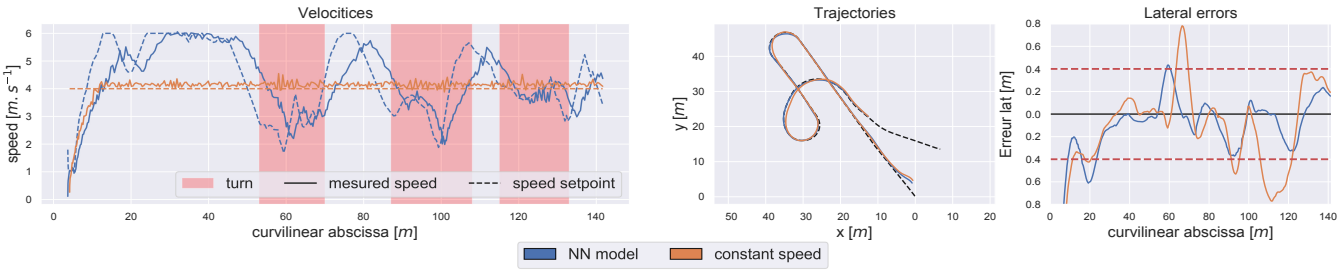

Fig. 7. Velocity and lateral error in the real world experiment

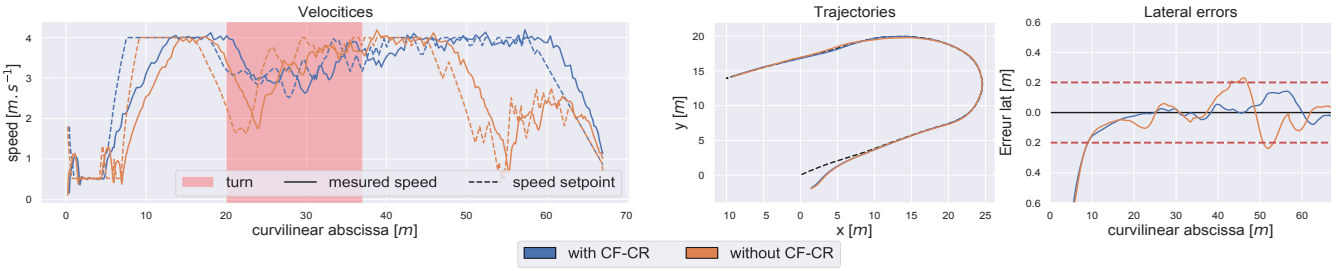

Fig. 8. Comparative plot between a neural network trained with and without cornering stiffnesses, in real world experiments

constant speed. With the neural network method however, the speed setpoint was reduced from $3.6\mathrm{m\,s^{-1}}$ to $1.9\mathrm{m\,s^{-1}}$ at $93\mathrm{m}$, while the steering controller reduced the steering angle to about $5°$. Both actions in tandem, allowed the system to converge back to a low lateral error with minimal perturbations. After the lateral error returned to acceptable values, the steering controller restored the original steer angle and the neural network method started a new acceleration phase. This modulation of the speed by the neural network minimized the error that was induced from the environment, allowing for a stable path following with minimal error.

In order to validate the importance of cornering stiffness inputs, a comparative experiment was done. The results of said experiment are displayed on Fig. 8. The neural network method that was trained without the cornering stiffnesses obtained considerably worse performance over the lateral error with a lower speed. From $s = 10$ to the end of the trajectory, the neural network with the cornering stiffnesses obtained a surface error of $2.77\mathrm{m}^2$, while the neural network without the cornering stiffnesses obtained a surface error of $4.89\mathrm{m}^2$, a 43.4% reduction in the surface error. This implies that the cornering stiffnesses are necessary in order to anticipate the future dynamic behavior of the next corner, and to slow down if necessary or to speed up if possible.

While a strict constraint over lateral error can be met in the simulated environment, in real application it is more difficult due to external disturbances such as GPS signal inaccuracies or terrain topology, as they can induce unmodeled lateral errors. Those disturbances cannot be anticipated and can be difficult to accurately model. As such this limits the reliability of the solution, and safety margins must be taken into account when defining the lateral error threshold.

High speed experiments have also shown unmodeled peaks in the lateral error that occurred in a deceleration phase during the corner (e.g. the spike at $s = 60m$). This implies some load transfer is occurring during the deceleration that is increasing the vertical force on the front wheels. The change in the vertical force causes a drastic change in the steering and the tyre grip, causing the system to react considerably faster, and as such causing oversteer which in turn produces higher error than those experienced in simulation.

V. CONCLUSIONS AND PERSPECTIVES

A reinforcement learning approach was proposed for speed fluctuation over a path following task. This method consists in training a neural network in order to control the acceleration for the speed setpoint to be as high as possible while maintaining the lateral error below a fixed threshold.

The neural network trained with a wide variety of paths, perception qualities and grip conditions was successful in setting the speed for both simulation and during real experiments. The simulator, based on a vehicle dynamic model, was able to train a model that is capable of obtaining favorable performance at high speed, in a real application without the use of transfer learning. This approach using a simple lateral error constraint can easily be extended to other hard constraints, such as a maximum angular deviation. Furthermore, the methodology is independent from the steering system and can be applied to other robot architectures with very little modifications.

However, some drawbacks appeared during the experiments. The longitudinal dynamic interactions may be considered in order to ensure an improved path following with less oversteering. And an improved steering controller designed to take into account speed variations, might improve the performance substantially.

Future works will consider a specific speed control for each wheel motor in order to increase the rotation speed of the system, similar to a previous deterministic approach [5]. Transfer learning techniques may also be involved to improve results by considering the real world wheel/ground dynamics. Merging this approach with previous work in online gain setting [33] is also being considered for future contributions.

REFERENCES

[1] T. Duckett, S. Pearson, S. Blackmore, and B. Grieve, "Agricultural robotics: The future of robotic agriculture," *CoRR*, vol. abs/1806.06762, 2018.

[2] S. Fountas, B. S. Blackmore, S. Vougioukas, L. Tang, C. G. Sørensen, and R. Jørgensen, "Decomposition of Agricultural tasks into Robotic Behaviours," *Agricultural Engineering*, p. 16, 2007.

[3] S. Blackmore, B. Stout, M. Wang, and B. Runov, "Robotic agriculture - the future of agricultural mechanisation?" *Precision Agriculture 2005, ECPA 2005*, p. 10, 2005.

[4] S. Bacha, M. Y. Ayad, R. Saadi, O. Kraa, A. Aboubou, and M. Hammoudi, "Autonomous Vehicle Path Tracking Using Nonlinear Steering Control and Input-Output State Feedback Linearization." in *2018 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM)*. Algiers: IEEE, Oct. 2018, pp. 1–6.

[5] R. Lenain, E. Lucet, C. Grand, B. Thuilot, and F. Ben Amar, "Accurate and stable mobile robot path tracking: An integrated solution for off-road and high speed context," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 196–201.

[6] F. Debrouwere, W. Van Loock, G. Pipeleers, Q. T. Dinh, M. Diehl, J. De Schutter, and J. Swevers, "Time-Optimal Path Following for Robots With Convex–Concave Constraints Using Sequential Convex Programming," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1485–1495, Dec. 2013, number: 6.

[7] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-Optimal Control of Robotic Manipulators Along Specified Paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, Sep. 1985, number: 3.

[8] T. Lipp and S. Boyd, "Minimum-time speed optimisation over a fixed path," *International Journal of Control*, vol. 87, no. 6, pp. 1297–1311, Jun. 2014, number: 6.

[9] S. Ebbesen, M. Salazar, P. Elbert, C. Bussi, and C. H. Onder, "Time-optimal Control Strategies for a Hybrid Electric Race Car," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 233–247, Jan. 2018, number: 1.

[10] F. Altche, P. Polack, and A. de La Fortelle, "High-speed trajectory planning for autonomous vehicles using a simple dynamic model," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. Yokohama: IEEE, Oct. 2017, pp. 1–7.

[11] C. Gámez Serna and Y. Ruichek, "Dynamic Speed Adaptation for Path Tracking Based on Curvature Information and Speed Limits," *Sensors*, vol. 17, no. 6, p. 1383, Jun. 2017, number: 6.

[12] M. Park, S. Lee, and W. Han, "Development of Steering Control System for Autonomous Vehicle Using Geometry-Based Path Tracking Algorithm," *ETRI Journal*, vol. 37, no. 3, pp. 617–625, Jun. 2015, number: 3.

[13] J.-B. Braconnier, R. Lenain, and B. Thuilot, "Ensuring path tracking stability of mobile robots in harsh conditions: An adaptive and predictive velocity control," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: IEEE, May 2014, pp. 5268–5273.

[14] J. Braconnier, R. Lenain, B. Thuilot, and V. Rousseau, "High speed path tracking application in harsh conditions: Predictive speed control to restrict the lateral deviation to some threshold," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3087–3094.

[15] O. Hach, R. Lenain, B. Thuilot, and P. Martinet, "Avoiding steering actuator saturation in off-road mobile robot path tracking via predictive velocity control," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Francisco, CA: IEEE, Sep. 2011, pp. 4072–4077.

[16] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[17] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, p. 359–366, Jul. 1989.

[18] J. Baltes and Y. Lin, "Path Tracking Control of Non-holonomic Car-Like Robot with Reinforcement Learning," in *RoboCup-99: Robot Soccer World Cup III*, G. Goos, J. Hartmanis, J. van Leeuwen, M. Veloso, E. Pagello, and H. Kitano, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, vol. 1856, pp. 162–173, series Title: Lecture Notes in Computer Science.

[19] L. Zhang, L. Qiao, J. Chen, and W. Zhang, "Neural-network-based reinforcement learning control for path following of underactuated ships," in *2016 35th Chinese Control Conference (CCC)*, 2016, pp. 5786–5791.

[20] D. Kamran, J. Zhu, and M. Lauer, "Learning Path Tracking for Real Car-like Mobile Robots From Simulation," in *2019 European Conference on Mobile Robots (ECMR)*. Prague, Czech Republic: IEEE, Sep. 2019, pp. 1–6.

[21] R. Lenain, B. Thuilot, C. Cariou, and P. Martinet, "Adaptive and Predictive Path Tracking Control for Off-road Mobile Robots," *European Journal of Control*, vol. 13, no. 4, pp. 419–439, Jan. 2007, number: 4.

[22] M. Deremetz, R. Lenain, B. Thuilot, and V. Rousseau, "Adaptive trajectory control of off-road mobile robots: A multi-model observer approach," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4407–4413.

[23] E. Bakker, L. Nyborg, and H. B. Pacejka, "Tyre modelling for use in vehicle dynamics studies," in *SAE Technical Paper*. JSTOR, 1987, pp. 190–204.

[24] E. Lucet, R. Lenain, and C. Grand, "Dynamic path tracking control of a vehicle on slippery terrain," *Control Engineering Practice*, vol. 42, pp. 60–73, 2015.

[25] A. Wolek, E. M. Cliff, and C. A. Woolsey, "Time-optimal path planning for a kinematic car with variable speed," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 10, pp. 2374–2390, 2016.

[26] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[28] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2015.

[29] A. Raffin and F. Stulp, "Generalized state-dependent exploration for deep reinforcement learning in robotics," *arXiv preprint arXiv:2005.05719*, 2020.

[30] D. Korenkevych, A. R. Mahmood, G. Vasan, and J. Bergstra, "Autoregressive policies for continuous control deep reinforcement learning," *arXiv preprint arXiv:1903.11524*, 2019.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[32] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[33] A. Hill, J. Laneurit, R. Lenain, and E. Lucet, "Online gain setting method for path tracking using cma-es: Application to off-road mobile robot control," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7697–7702.