2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Online gain setting method for path tracking using CMA-ES: Application to off-road mobile robot control

Ashley Hill[1], Jean Laneurit[2], Roland Lenain[2] and Eric Lucet[1]

***Abstract*— This paper proposes a new approach for online control law gains adaptation, through the use of neural networks and the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm, in order to optimize the behavior of the robot with respect to an objective function. The neural network considered takes as input the current observed state as well as its uncertainty, and provides as output the control law gains. It is trained, using the CMA-ES algorithm, on a simulator reproducing the vehicle dynamics. Then, it is tested in real conditions on an agricultural mobile robot at different speeds. The transferability of this method from simulation to a real system is demonstrated, as well as its robustness to environmental changes, such as GPS signal degradation or ground variation. As a result, path following errors are reduced, while ensuring tracking stability.**

I. Introduction

In mobile robotics, accuracy and stability are usually desired qualities of a controller, especially in the framework of path tracking. They depend, however, on the correct tuning of the controller gains, which are generally tuned empirically by an expert or with the help of dedicated algorithms such as the Ziegler–Nichols method.

Achieving this goal with a constant gain is not obvious, as the optimal gain depends on multiple environmental conditions, such as the quality of perception, ground sliding conditions [Samson et al., 2016], and also the robot's speed. A constant gain for different configurations and contexts of navigation is therefore not ideal. As such, an optimal gain will need to be adaptive in order to maximize the performance of the controller at any given time.

Methods for adaptive gains exists in the current state of the art, such as fuzzy logic gain scheduling [Khesrani et al., 2017], or LQR methods for gain tuning [Argentim et al., 2013]. However fuzzy logic gain scheduling requires an expert to tune a constant gain for each environmental case, meanings the gains will be sub optimal for the edges of each case. And using an LQR for gain tuning, implies finding the optimal gains for said LQR, offsetting the problem.

Thus, the work presented in this paper is based on extending the previous state of the art on adaptive gains without the use of an expert for gain tuning, with real experimentation on an off-road mobile platform, with a complex and realistic environment.

For this purpose, gain adapting will be defined as predicting the quasi-optimal gain, from changes in the environment indicated in the environmental state, which are encoded in the observed state. This involves determining an approximate function, capable of mapping the observed state to the near-optimal gain.

Several works on this have been proposed, specifically the gain adaptation in real time using neural networks, such as using a gradient based approach [Guo et al., 2009], [Omatu and Yoshioka, 1997], [Chang et al., 2003] or with an evolution based approach [Hill. et al., 2019], [Omatu and Yoshioka, 1997].

However, these works are done with either: simple robot dynamics, simple controllers, constant environments, or no real experimentation. Furthermore, the performance of the gradient methods is sub-optimal, as they require an exact and noiseless model of the robotic system. As such, the methods of this paper is based on an evolutionary approach, specifically from [Hill. et al., 2019] as they have done similar experimentation, with good results in simulation.

In this paper, the simulation that was used to train the method take into account: a tire slip model, a steering delay model, a dynamic model, noisy observations, and controllers used in agricultural environment [Cariou et al., 2008], [Lenain et al., 2007]. This was done in order to reach a more realistic simulation of the real world experiments, with controllers that are optimized for the task, so that the model could work as is with the robot, and that a good baseline comparison with a constant gain method could be obtained.

The details of the extensions to the simulation and the robotic context are detailed in section II, with the proposed method detailed in section III, followed by the training setup and simulated results in section IV, with the experimental setup detailed in section V, followed by real world experimentation in section VI, ending with a discussion of the results and some perspectives in section VII.

II. Robotic model & context

The robotic task is path following of a known trajectory, using a Extended Kalman filter due to it's ubiquity. This task will be considered in a agricultural context, due to the highly dynamic environments, with lower speeds. The robot used is a $500kg$, 4 wheeled mobile robot, with front steering.

In the following subsections, the robotic model and the control will be detailed.

A. Controller

In order to test the proposed method, the following controllers were used. The first is called *Classic* which is defined in the paper [Cariou et al., 2008], and is based on an extended

[1]CEA, LIST, Interactive Robotics Laboratory, Gif-sur-Yvette, F-91191, France firstname.lastname@cea.fr
[2]Université Clermont Auvergne, Inrae, UR TSCF, Centre de Clermont-Ferrand, F-63178 Aubière, France firstname.lastname@inrae.fr

kinematic model, including the sideslip angles β_F and β_R, which are estimated online using an observer, detailed in the mentioned reference [Lenain et al., 2017]. The second is a variant of the *Classic* controller, but with predictive control, it is called the *Predictive* controller, and it is defined in the paper [Lenain et al., 2007].

Both controllers are based on the same control equation, which computes the steering angle values, it can be defined as following:

$$\delta_F = \tan^{-1}\left\{\tan(\beta_R) + \frac{L}{\cos(\beta_R)}\left(\frac{c(s)\cos\tilde{\theta}_2}{\alpha} + \frac{A\cos^3\tilde{\theta}_2}{\alpha^2}\right)\right\} + \beta_F \tag{1}$$

with:

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

These controllers take two gains as in order to tune them:

- K_p: which defines the distance to convergence with respect to the lateral error.
- K_d: which defines the distance to convergence with respect to the angular error.

And the variables used for its computation are defined as follows and depicted on the Fig. 1: y The tracking error, $\tilde{\theta}$ the angular deviation, $c(s)$ the curvature at the curvilinear abscissa s, β_F and β_R the front and rear sliding angle respectively.

B. Simulation

In order to train the proposed method, a simulation must be used for both time and safety reasons. For the robotic model, a dynamic bicycle model is used (shown Fig. 1). With a

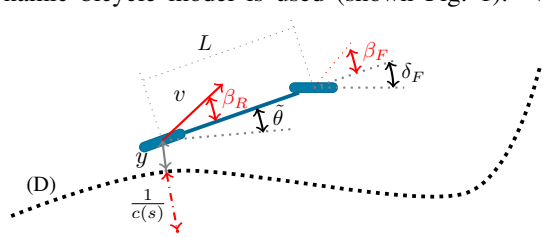

Fig. 1. The mobile robot studied.

pure action delay of $0.5ms$ on the steering, as it is the worst case steering action delay for the Adap2e platform [Deremetz et al., 2019], that is used in the experimentation. The dynamic simulation took into account the mass of the robot ($500kg$), the vertical inertial ($400kg.m^2$), and a Pacejka tire slip model [Bakker et al., 1987] with gravel-like sliding conditions. The model was also trained with a varying speed of $1.0ms^{-1}$, $1.5ms^{-1}$, and $2.0ms^{-1}$. The simulator was written in C++ and Python, for ease of use and high performance.

C. Objective functions

In order to train the method, a goal must be defined and quantified in an equation; where the global minimum of this equation is considered the ideal performance. For this, an objective function ob_1 is defined (equation 2). It has been chosen in order to minimize the tracking error, decrease the oscillation, while limiting important actions on the steering angle. This objective function takes into account both the path tracking errors and the steering energy.

$$ob_1 = \frac{1}{T}\sum_{n=0}^{N}\left[|y(t_n)| + L|\tilde{\theta}(t_n)| + 0.5L|\delta_F(t_n)|\right] dt \quad [m] \tag{2}$$

Where T is the total time taken to follow the path, N is the number of measured timesteps, t_n is the time at the timestep n, y is the lateral error along the path, L is the wheelbase length, $\tilde{\theta}$ is the angular error along the path, δ_F is the front steering angle, and dt is the time between each timestep. The value of 0.5 has been set as the weighting coefficient for the steering energy, this value was found empirically.

III. Proposed method

The online controller gains optimization proposed is based on the control scheme detailed on the Fig. 2. As it is depicted on this figure, an adaptive control loop (in blue) is used to achieve the path tracking task. The controller (detailed in the section II-A) is tuned with two gains, defining the settling distance, and the damping coefficient, classically defined as constants. Such constant gains are defined in order to ensure the stability of the path tracking with expected conditions (speed, grip conditions, sensor noise...), and as such appears to be non-optimal, when those conditions change over time.

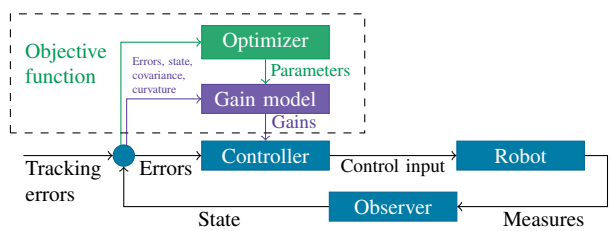

Fig. 2. Overview of the proposed method, with the contribution in the dashed rectangle.

To overcome this drawback, a method dedicated to computing online optimal gains is introduced, based on a gain model (in purple on the Fig. 2). This gain model is fed with the tracking variables (such as tracking error, angular deviation, curvature, sensor noise, ...), and is designed to minimize an objective function, using an optimizer (depicted by the green block). The optimizer proposed here is Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Hansen, 2016], due to its performance in complex search spaces such as neural networks [Hansen et al., 2010]. The gain model being a neural network, due to its capability to be a universal approximator [Hornik et al., 1990], with hyperbolic tangent as its activation function and with hidden layers of 40, 100 and 10. This was chosen in order to reach a non linear transformation, while having a low number of parameters so the CMA-ES method can perform optimally.

In order to achieve online computing of the optimal gains, the neural network is trained off-line in a simulation, using the CMA-ES optimizer guided by the objective function; as exploring the gain space is time consuming, and can lead to dangerous or unstable behavior in the real world conditions.

This global scheme has been previously introduced in [Hill. et al., 2019] for a single gain adaptation, showing promising results in simulation, with a simple robotic model. In this paper, a whole algorithm is proposed and tested through full scale experiments.

Classically, machine learning methods train to minimize a target metric (here the objective function), however measuring the performance with the objective function must be avoided, as the method might exploit the target metric, at the detriment to other metrics (classically called reward shaping). As such, to compare and interpret the quality of the trajectory tracking performance, the following metric is used:

$$A_{\text{error}} = \sum_{n=0}^{N} \left| \dot{s}(t_n) \left(y(t_n) + \frac{\dot{y}(t_n)\, dt}{2} \right) \right| dt \qquad [m^2] \quad (3)$$

with:

$$\begin{cases} \dot{s}(t) &= v(t) cos(\tilde{\theta}(t)) \\ \dot{y}(t) &= v(t) sin(\tilde{\theta}(t)) \end{cases}$$

Where t_n is the time at the timestep n, s is the curvilinear abscissa of the trajectory, y is the lateral error along the trajectory, dt is the sampling rate of the robot's control (set to $0.1s$ in this context), $\tilde{\theta}$ is the angular error along the trajectory, and $\dot{s}$, $\dot{y}$ are the derivative of s, y respectively over the time t.

The equation (3) is the approximate integration of the surface error between the robot's position and the target trajectory using a Trapezoidal Riemann sum. This allows comparison of performance between gain tuning methods, independently of speed and time for the given trajectory.

IV. Training setup

Two methods will be tested and compared in this paper:

- *Expert constant gain*: The expert tuned constant gain for the robot defined for each experiment.
- *NN* ob_1: the proposed method, trained with the ob_1 objective function defined in equation (2).

The baseline method *Expert constant gain* is set to a constant value, given by an expert who has tuned the gains for the robotic platform and the controllers used.

The proposed method *NN* ob_1 however, need to be trained in order to explore the possible gain space, and determine the optimal gains for the robot and controller.

The training is done in simulation, as the CMA-ES methods is too sample inefficient in order to be run in real world condition in a reasonable time. It has be set with a population size of 32, an initial sigma of 0.04, and 20000 maximum evaluations as its time budget. The *Classic* controller is used for the training. The coefficients of the Pacejka sliding curve, were defined as to emulate poor sliding conditions. And an action delay of $500ms$ was defined. The training was done over each trajectory 5 times, at 3 speeds $1.0m.s^{-1}$, $1.5m.s^{-1}$, and $2.0m.s^{-1}$. It was trained with 2 trajectories called *sine*, a sinusoidal curve and *spline0*, a sharp "S" curve. Midway in the trajectories, a GPS noise of $1m$ is applied to simulate a loss of GPS signal. These trajectories were chosen in order to simulate difficult paths for the Adap2e platform and the controllers to follow accurately, with some simpler sections in each trajectories so the proposed model does not over-fit for complex trajectories.

A. Simulation results

The method having been trained in simulation, it suffers from the simulation's systematic error and inaccuracies with reality, and as such the following simulated experiments will show the ideal theoretical performance of the method.

The proposed control scheme has been applied for the previously defined trajectories *sine* and *spline0*. And the performance obtained can be observe in Fig. 3. Using the A_{error} metric defined in equation (3), the performance of the proposed method is compared with the baseline method (i.e. constant gain fixed by an expert for a speed of $2.0m.s^{-1}$), in order to get a percentage and absolute improvement in the surface of the lateral error. A substantial decrease in the

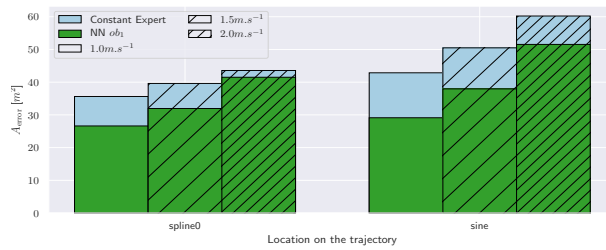

Fig. 3. The A_{error} of the tested method and the *expert constant gain*. The hash density show the changes in the speed.

surface error can be observed for the *NN* ob_1 method (with an average decrease of 20%). However, the performance gap seems reduced at $2.0m.s^{-1}$, as this is the optimal speed for the expert gain.

This implies that during the experimental results, that the *NN* ob_1 might out perform the baseline method. But only if the method is robust enough to transfer from the simulation to real world conditions, which is classically a difficult problem for machine learning methods.

V. Experimental setup

Experiments were performed using the Adap2e platform [Deremetz et al., 2019] that can be seen on Fig. 4. It is a mobile robot dedicated to carrying tools for agricultural operations, capable of adapting to the diversity of tasks and parcels of land. Driven in a car-like configuration with only the front wheels steered, it has a track width of $1.0m$, and a wheel base of $1.38m$. Software developments are carried out on the ROS middelware. The control and GPS update rate are $0.1s$. Tests were carried out on a warm, sunny day with cloudless weather, meaning very dry soil and low GPS perturbation.

The trajectory shown in Fig. 4 is first on a ground made up of gravel and earth. Then, near the beginning of the turn and until the end of the trajectory, the ground is grass. The terrain also has an irregular topology. The robot starts with an initial lateral error of approximately $1m$, relative to a straight line trajectory, in order to observe its convergence towards this trajectory. Then, a half turn that is wide enough so as not

Fig. 4. On the left: The Adap2e robot. On the right: The trajectory over the ground.

to saturate the steering actuator must be followed. Finally, a final straight line is followed after exiting the turn. This trajectory is displayed in Fig. 5.

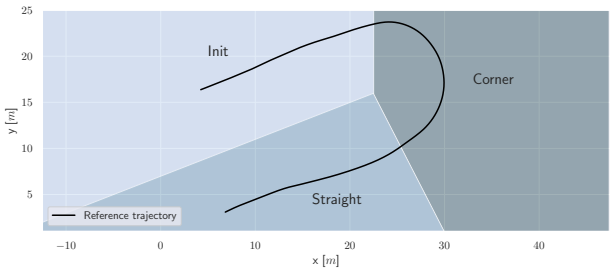

Fig. 5. The reference trajectory on an x,y scale.

This trajectory has been used to validate the proposed method against overfitting, meaning a trajectory not previously used for the learning process in order to show the genericity of the trained solution with regard to path shapes. It is broken down into three parts of $20m$ each for the analysis :

- *Init*: The start of the trajectory, the robot is launched from approximately $1m$ from the side of the trajectory as an initial error.
- *Corner*: The large corner of the trajectory, with a constant curvature of $0.2m^{-1}$.
- *Straight*: The final straight line after the large corner, in order to observe the stabilization from the corner.

Results are also compared over the entire $60m$ of the trajectory (defined as *Total*), in order to get a general idea of the expected improvements.

VI. Results

The experiments of this section are divided into three sets, the first using the *classic* controller, the second using the *predictive* controller, and the third is used to observe the reaction of the proposed method to changes in the environmental state, by adding a Gaussian noise of $1m$ to the GPS signal. This is done in order to investigate the performances of the proposed algorithm in situations with high uncertainty.

A. Experiment 1

The first experiment consists of using the *classic* controller without any sliding observers (β_F and β_R are set to zero in the control expression (1)), using the proposed method *NN* ob_1 and the baseline *expert constant gain* method. The *expert constant gain* was set to $K_d = 0.7,\ K_p = 0.1225$, which is tuned for the robot considering a constant speed of $2.0m.s^{-1}$. The experiment was run with the target speeds of $1.0m.s^{-1}$, $1.5m.s^{-1}$, and $2.0m.s^{-1}$. Comparing the proposed method *NN* ob_1, with the baseline *expert constant gain* method, the following analysis is obtained.

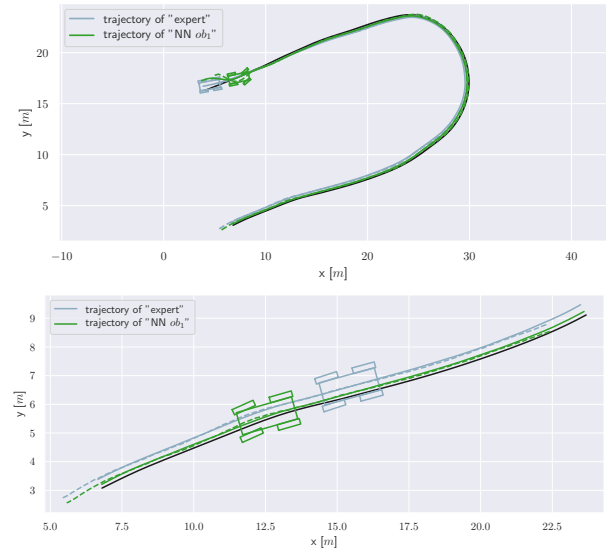

Fig. 6. The estimated paths traces by each method is set at the solid line for the rear axle, and in the dashed line the front axle. Above: the path over the full trajectory. Below: the path over the *Straight* segment.

The paths traced on Fig 6, show the trajectories of the robot for each method at $1.0m.s^{-1}$. From this, the proposed method *NN* ob_1 is able to more closely follow the trajectory overall, and specifically near the end of the trajectory.

On Fig. 7, using the A_{error} metric defined in equation (3), the performance of the proposed method is compared with the baseline method, in order to get a percentage and absolute improvement in the surface of the lateral error. A substantial

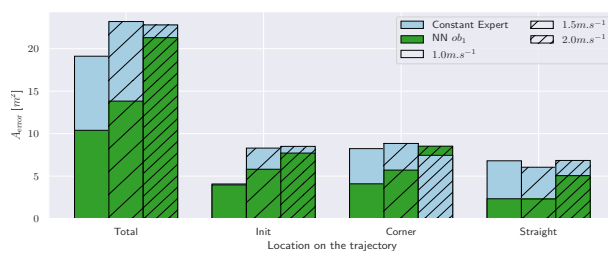

Fig. 7. The A_{error} of the tested method and the *expert constant gain*. The hash density show the changes in the speed.

increase in the accuracy can observed between the proposed method and the baseline method over a target speed of $1.0m.s^{-1}$ (44%) and $1.5m.s^{-1}$ (39%), with still minor improvements to the performance at $2.0m.s^{-1}$ (5%). It is important to note, the baseline method started with $0.2m$ of initial error for the target speed of $1.0m.s^{-1}$ (visible on Fig 6), hence the low improvement in the *Init* part of the trajectory for that speed.

These are similar to the results obtained in simulation, with a significant reduction in the overall error. They show that in real world conditions, the modulation of the gain in real time, can allow for higher gains in certain circumstances, which

leads to a more optimal path tracking, without inducing instabilities.

After an initial proof of concept with a sufficiently simple *classic* controller, a *predictive* controller more suitable to autonomous mobile robots is considered for experiment 2, by using sliding observers defined in [Lenain et al., 2017].

B. Experiment 2

The second experiment consists of using the *predictive* controller with sliding observers defined in [Lenain et al., 2017], along with the proposed method *NN* ob_1 and the baseline *expert constant gain* method. The *expert constant gain* was set to $K_d = 0.7,\ K_p = 0.1225$. The experiment was run with the target speeds of $1.0m.s^{-1}$, $1.5m.s^{-1}$, and $2.0m.s^{-1}$. Comparing the proposed method *NN* ob_1, with the baseline *expert constant gain* method, the following analysis is obtained.

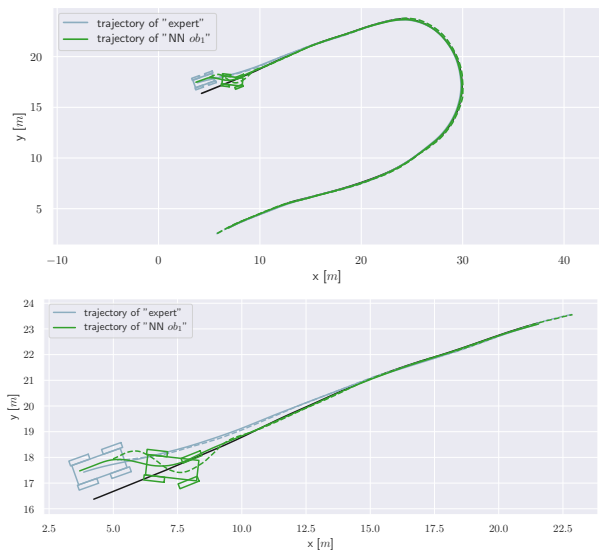

Fig. 8. The estimated paths traces by each method is set at the solid line for the rear axle, and in the dashed line the front axle. Above: the path over the full trajectory. Below: the path over the *Init* segment.

The paths traced on Fig 8, show the trajectories of the robot for each method at $1.0m.s^{-1}$. From this, the adaptability of the proposed method *NN* ob_1 can be see, as it is able to quickly correct the initial error without overshooting with the robot's rear axle.

On Fig. 9, using the A_{error} metric defined in equation (3), the performance of the proposed method is compared with the baseline method, in order to get a percentage and absolute improvement in the surface of the lateral error.

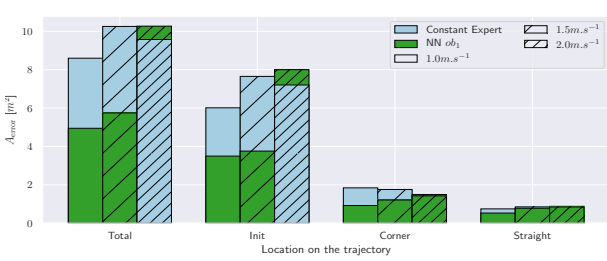

Fig. 9. The A_{error} of the tested method and the *expert constant gain*. The hash density show the changes in the speed.

Similarly to experiment 1, the proposed method is substantially more accurate than the baseline method over a target speed of $1.0m.s^{-1}$ (40%) and $1.5m.s^{-1}$ (42%). However with minor degradation in the performance at $2.0m.s^{-1}$ (-7%), since the constant gain was tuned for a speed of $2.0m.s^{-1}$.

The loss of performance at $2.0m.s^{-1}$ is due to the proposed method being trained in a simulation, and faces here the side effects of the systematic error. Indeed as the speed increases, the dynamic effects start influencing the robot's trajectory and velocity in a stronger manner. Furthermore, the model was not given the dynamic parameters (such as the sliding angles) as an input during the simulation, this means it was not trained to handle the change in the gain relative to varying sliding conditions.

So far the experiments have shown some of the benefits and drawbacks of the proposed method in a semi-static environment state. However, this method should adapt the gain with respect to strong variations in the environment state. As such, the experiment 3 consists of varying the quality of the GPS perception.

C. Experiment 3

The third experiment consists of using the *predictive* controller with sliding observers defined in [Lenain et al., 2017], along with the proposed method *NN* ob_1 and the baseline *expert constant gain* method. The *expert constant gain* was set to $K_d = 1.0,\ K_p = 0.25$, which is tuned for the robot with the trajectory for a speed of $1.0m.s^{-1}$. The GPS signal degradation starts midway in *Corner* part of the trajectory, and stays to the end. In order to make the results comparable, the Gaussian random number generator was set with the same seed each time, this means the noise was repeatable and consistent between each test of the experiment. The experiment was run with the target speed of $1.0m.s^{-1}$. Comparing the proposed method *NN* ob_1, with the baseline *expert constant gain* method, the following analysis is obtained.

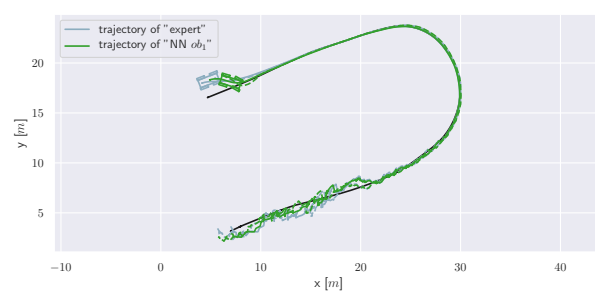

Fig. 10. The estimated paths traces by each method is set at the solid line for the rear axle, and in the dashed line the front axle over the full trajectory.

The paths traced on Fig 10, show the trajectories of the robot for each method at $1.0m.s^{-1}$. From this, the behavior of the robot cannot be ascertained, due to the GPS noise clouding the estimated paths. As such, only the quantitative results can reflect any changes in behavior.

On Fig. 11, using the A_{error} metric defined in equation (3), the performance of the proposed method is compared with

the baseline method, in order to get a percentage and absolute improvement in the surface of the lateral error. For the

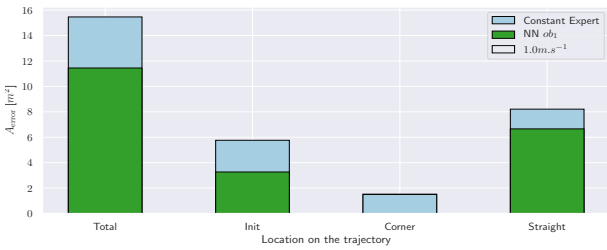

Fig. 11. The A_{error} of the tested method and the *expert constant gain*.

surface error A_{error}, the proposed method *NN* ob_1 was able to substantially reduce the error overall (25%) and in the *Straight* part of the trajectory (20%), despite an important loss of accuracy in the localization. From this, the proposed method *NN* ob_1 is increasing the dampening of the controller, in order to avoid the controller reacting too strongly to the input noise. This allows the proposed method and controller to more accurately follow the trajectory, even in very noisy conditions.

The results obtained in the experiments seem to concord with the simulation results, but diverging slightly at $2.0m.s^{-1}$ due to the higher systematic error, causing the *NN* ob_1 model to experience a drop in performance when compared to the simulated performance. This systematic error might due to the poorly characterized dynamic parameters, which are very difficult to accurately estimate for a given robot.

VII. Conclusions and discussion

An algorithm for online adjusting of gains was proposed, based on neural networks trained from a simulated dynamic model. This method was applied to the tuning of gains of a steering angle controller, for the optimization of the robot's behavior in relation to lateral displacements. It takes into account the target speed, path curvature, lateral and angular tracking errors, and perception quality (state observer covariance matrix), in order to predict near optimal sets of gains that minimize an objective function. Therefore, the robot settling distance, as well as the tracking error are minimized, while avoiding instabilities.

The proposed method was tested in real conditions without any transfer learning or adaptation. Indeed, the errors in modeling the system dynamics are sufficiently small at the speed of testing to limit deviations from reality.

Experimental tests were conducted at different speed values, and by considering GPS signal losses. Results show an increase in the accuracy of trajectory tracking compared to constant gain methods, in some cases up to 44%.

However, this method has some drawbacks. First, a long training time in simulation is due to the nature of the CMA-ES optimization algorithm that needs a large amount of trajectory following simulations. Then, it becomes increasingly sensitive to dynamic model errors at higher speeds. Indeed, gain computation should be done with consideration of this dynamics, in particular observed sideslip angles β_F and β_R. Therefore, such information can be used as input to the neural network in order to adapt the robot's behavior to grip conditions. Finally, longitudinal speed has also a significant impact on the robot's behavior. As a result, future work is focused on the consideration of sideslip angles and longitudinal speed for gain tuning.

Going further, the online optimization of the target longitudinal speed, depending on the grip conditions and of the shape of the desired trajectory, is an area of investigation.

VIII. Acknowledgments

This publication was made possible by the use of Factory-IA cluster, financially supported by the Ile-de-France Regional Council.

References

[Argentim et al., 2013] Argentim, L. M., Rezende, W. C., Santos, P. E., and Aguiar, R. A. (2013). Pid, lqr and lqr-pid on a quadcopter platform. In *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 1–6. IEEE.

[Bakker et al., 1987] Bakker, E., Nyborg, L., and Pacejka, H. B. (1987). Tyre modelling for use in vehicle dynamics studies. Technical report, SAE Technical Paper.

[Cariou et al., 2008] Cariou, C., Lenain, R., Thuilot, B., and Martinet, P. (2008). Adaptive control of four-wheel-steering off-road mobile robots: Application to path tracking and heading control in presence of sliding. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1759–1764. IEEE.

[Chang et al., 2003] Chang, W.-D., Hwang, R.-C., and Hsieh, J.-G. (2003). A multivariable on-line adaptive pid controller using auto-tuning neurons. *Engineering Applications of Artificial Intelligence*, 16(1):57–63.

[Deremetz et al., 2019] Deremetz, M., Couvent, A., Lenain, R., Thuilot, B., and Cariou, C. (2019). A generic control framework for mobile robots edge following. In *International Conference on Informatics in Control, Automation and Robotics*.

[Guo et al., 2009] Guo, B., Liu, H., Luo, Z., and Chen, W. (2009). Adaptive pid controller based on bp neural network. *2009 International Joint Conference on Artificial Intelligence*, pages 148–150.

[Hansen, 2016] Hansen, N. (2016). The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772.

[Hansen et al., 2010] Hansen, N., Auger, A., Ros, R., Finck, S., and Pošík, P. (2010). Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, pages 1689–1696, New York, NY, USA. ACM.

[Hill. et al., 2019] Hill., A., Lucet., E., and Lenain., R. (2019). Neuroevolution with cma-es for real-time gain tuning of a car-like robot controller. In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,*, pages 311–319. INSTICC, SciTePress.

[Hornik et al., 1990] Hornik, K., Stinchcombe, M., and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551 – 560.

[Khesrani et al., 2017] Khesrani, S., Hassam, A., Boubezoula, M., and Srairi, F. (2017). Modeling and control of mobile platform using flatness-fuzzy based approach with gains adjustment. In *2017 6th International Conference on Systems and Control (ICSC)*, pages 173–177. IEEE.

[Lenain et al., 2017] Lenain, R., Deremetz, M., Braconnier, J.-B., Thuilot, B., and Rousseau, V. (2017). Robust sideslip angles observer for accurate off-road path tracking control. *Advanced Robotics*, 31(9):453–467.

[Lenain et al., 2007] Lenain, R., Thuilot, B., Cariou, C., and Martinet, P. (2007). Adaptive and predictive path tracking control for off-road mobile robots. *European journal of control*, 13(4):419–439.

[Omatu and Yoshioka, 1997] Omatu, S. and Yoshioka, M. (1997). Self-tuning neuro-pid control and applications. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 3, pages 1985–1989. IEEE.

[Samson et al., 2016] Samson, C., Morin, P., and Lenain, R. (2016). Modeling and control of wheeled mobile robots. In *Springer Handbook of Robotics*, pages 1235–1266. Springer.