Adaptive Nonlinear Guidance Law Using Neural Networks Applied to a Quadrotor

Bartomeu Rubí[1]* and Bernardo Morcego[1] and Ramon Peréz[1]

***Abstract*— The *NonLinear Guidance Law* (*NLGL*) is a geometric algorithm commonly employed to solve the path following problem on different unmanned vehicles. *NLGL* is simple (does no depend on the model of the vehicle), effective and has only one tunning parameter. Its control parameter (L) depends on various factors, such as the velocity of the vehicle, the shape of the reference path and the dynamics of the vehicle.**

This paper analyses the effect of parameter L on the performance of *NLGL* when it is applied to a quadrotor vehicle. An *Adaptive NLGL*, which includes a velocity reduction term, is proposed. Stability proofs are given. Simulation results show that the proposed algorithm enhances the performance of the standard *NLGL*. Furthermore, it has no parameters to tune.

I. INTRODUCTION

Multirotor vehicles highlight for their good maneuverability, stability and fast performance as well as their high payload and fault tolerant capabilities. Nowadays, the research in this type of Unmanned Aerial Vehicles (UAVs) is mainly focused on trajectory control, take off and landing, obstacle avoidance and path planning problems.

Trajectory control, consisting on making a vehicle follow a given trajectory, can be defined by two different problems: trajectory tracking and path following. In the trajectory tracking problem a position reference that varies in time is tracked. The control objective of the path following problem is to make the vehicle converge to a given path without any time specification. Furthermore, the approach of the path following problem results in many advantages [1][2][3] such as smoother path convergence, less control effort demand, smaller transient error and a stronger robustness.

Geometric guidance algorithms are commonly employed to solve the path following problem. These algorithms were originally described in the missile guidance literature, but most of them were adapted to other type of vehicles, such as UGVs [4][5] or UAVs. Examples of geometric algorithms that are applied to UAVs are: *NonLinear Guidance Law* (*NLGL*) [6], *Trajectory Shaping* [7], *Vector-field* based [8] and *Pure Pursuit* [9].

Some of those geometric guidance algorithms have very few (1 or 2) parameters to tune. These parameters define the performance and the stability of the controller [6][5]. The choice of the proper values of those parameters depend on factors such as the velocity of the vehicle or the reference path shape, and needs to be done manually each time the experiments conditions vary. In [10] the authors adjust these parameters for different path following algorithms by means of an optimization procedure based on *Genetic Algorithm* technique. However, this optimization is performed off-line and needs to be redone when path conditions vary. The present paper is focused on the study of the parameter selection for the *NonLinear Guidance Law*, and presents an adaptive approach based on the use of a neural network. That is, the parameters of *NLGL* are automatically generated (on-line) depending on the path shape and the vehicle's velocity. Stability proofs of the proposed approach are given. Finally, the performance of the adaptive approach is assessed by numerical simulations.

*Corresponding author

[1] Authors are with the Specific Center of Research at CS2AC in the Universitat Politècnica de Catalunya (UPC). Rbla Sant Nebridi 22, Terrassa (Spain). email: tomeu.rubi, bernardo.morcego, ramon.perez at upc.edu

II. PROBLEM STATEMENT

The aim of this paper is to develop an adaptive strategy for the *NonLinear Guidance Law* (*NLGL*) applied to a quadrotor vehicle to solve the path following problem.

A. Quadrotor model

The quadrotor dynamic model employed in this paper is a standard nonlinear model, described in detail in literature [11][12]. The frames of reference (world frame $\{W\}$, considered inertial, and body frame $\{B\}$, attached to the vehicle) and other conventions of the quadrotor model are presented in Fig. 1. This model has twelve states: the body velocities (u, v and w), the world position (x, y and z), the body angular velocities (p, q and r) and euler angles (ϕ-*roll*, θ-*pitch* and ψ-*yaw*). And it has four inputs: thrust force (T) and torque moments applied to each axis (τ_ϕ, τ_θ and τ_ψ).

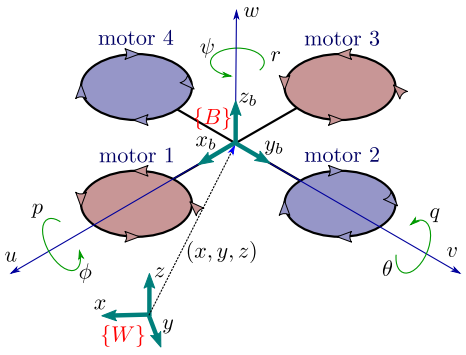

Fig. 1. States conventions and frames of references of the quadrotor model.

B. Path following problem

The path following problem is defined as making the quadrotor follow a pre-stablished path in space. The reference path can be given using different approaches such

as with waypoints [13][8], dubins paths [14][15] or splines [16][17]. In this paper, a parametrized function of the three position coordinates is used to define the desired path:

$$\mathbf{p}_d(\gamma) := [x_d(\gamma), y_d(\gamma), z_d(\gamma)]^T \tag{1}$$

where γ is the virtual arc that parametrizes the curve.

C. NonLinear Guidance Law

The *NonLinear Guidance Law* (*NLGL*) [18] is a geometric algorithm based on the notion of the virtual target point (VTP). That is, a point placed on the reference path which the vehicle is required to follow. If the vehicle is steered towards the VTP, it will end up converging to the path and following it. In *NLGL*, the VTP is obtained as the point on the path that is at a distance *L* from the vehicle. L is a scalar parameter that defines the path following performance, that is, the parameter that will be analyzed in this paper.

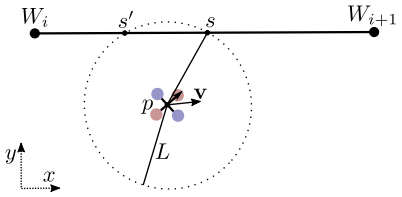

Fig. 2. Obtaining the VTP for the NLGL for a straight line path.

Fig. 2 shows how the VTP can be obtained for a 2D straight path between two waypoints (W_i and W_{i+1}). In this case, a circumference of radius L is created around the vehicle. Two intersection points of the circumference with the path are obtained (s and $s\prime$). The closest point to the next waypoint (s) is the VTP. Next, velocity and attitude commands are generated to steer the vehicle to that VTP. Then, these commands are tracked by an attitude controller, commonly known as the autopilot.

To obtain the VTP for the general case of a 3D curve defined as in Eq. (1), first, the minimum distance between the vehicle and the path, d_{min}, is calculated (Eq. (2)). The point on the path at minimum distance from the vehicle is also obtained, Eq. (3). If d_{min} is longer than L means that the vehicle is still too far from the path, thus, the VTP is chosen as the point on the path closest to the vehicle.

$$d_{min} := \min_{\gamma} \left(\|\mathbf{p} - \mathbf{p}_d(\gamma)\| \right) \tag{2}$$

$$\gamma_{d_{min}} := \operatorname*{argmin}_{\gamma} \left(\|\mathbf{p} - \mathbf{p}_d(\gamma)\| \right) \tag{3}$$

If d_{min} is smaller or equal than L, the VTP is obtained as

$$\gamma_{VTP} := \arg_{\gamma} \left(|\|\mathbf{p} - \mathbf{p}_d(\gamma)\| - L| = 0 \right). \tag{4}$$

In case Eq. (4) leads to more than one solution, the closest one to $\gamma_{d_{min}}$ is chosen, checking that $\gamma_{VTP} > \gamma_{d_{min}}$ to make the vehicle always move forward on the path.

D. Control Structure

The control structure employed in this paper is presented in Fig. 3. It is a Separated Guidance and Control (SGC) structure [19], since the translational dynamics and the rigid-body rotational dynamics are controlled by two different control modules: the path following controller and the autopilot. The path following controller implements the guidance law. The autopilot consists of a velocity controller and an attitude controller that generate the inputs to the quadrotor (T, τ_ϕ, τ_θ, τ_ψ). The output of the quadrotor ($\mathbf{x}$) corresponds to the full state of the vehicle (position, velocity and angular states).

The path following controller receives the path reference ($\mathbf{p}_d(\gamma)$) and the vehicle's position (x, y, z) and *yaw* (ψ) states. It generates the references for altitude (z_{cmd}), *yaw* (ψ_{cmd}) and lateral body velocities (v_{cmd} and u_{cmd}). ψ_{cmd} is the main output of the path following algorithm and it is used as the control command to steer the vehicle. The longitudinal velocity u_{cmd} is obtained as

$$u_{cmd} := V_{ref} \frac{d_{xy,VTP}}{L}, \tag{5}$$

where $d_{xy,VTP}$ is the distance from the vehicle to the VTP on the xy plane and V_{ref} is the velocity reference.

The state-of-art *NLGL* is a 2D algorithm. However, it can be translated to the 3D space by setting the value of the path altitude on $\gamma_{d_{min}}$ to the vehicle's altitude command ($z_{cmd} = z(\gamma_{d_{min}})$) and assuring that the velocity on the x axis, u_{cmd}, decreases when the vehicle is required to ascend or descend. That is, the velocity reference is reduced when the path is not on the horizontal plane, Eq. (5).

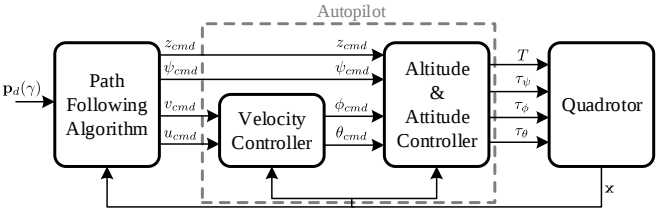

Fig. 3. Control structure for geometric algorithms.

All the results and graphs reported in this paper were obtained with the Path-Flyer quadrotor simulator [20], which implements the proposed control structure along with the standard nonlinear dynamic equations. It also includes the dynamics of the motors and other aerodynamic effects, such as drag forces. In this simulator the autopilot is implemented by a set of PID controllers.

III. PERFORMANCE OF *NLGL* DEPENDING ON THE PARAMETER SELECTION

This section analyzes the performance of *NLGL* as a function of parameter L. The mean absolute error (MAE) in terms of distance to the path, noted $\overline{d_{min}}$ hereafter, is used to evaluate the performance of the algorithm. The dynamics of the autopilot and the vehicle are not modified throughout the paper. That is, it is not analyzed how having different inner dynamics affects on the optimal selection of L.

A. Path distance MAE in function of parameter L

The simulation scenario is a circular path of radius R and a constant velocity reference of $1^m/_s$. The vehicle starts in a point on the path at hover condition (i.e. zero linear and angular velocities) and it is requested to perform a full lap of the path. The average distance of the vehicle to the path ($\overline{d_{min}}$) is measured in each simulation.

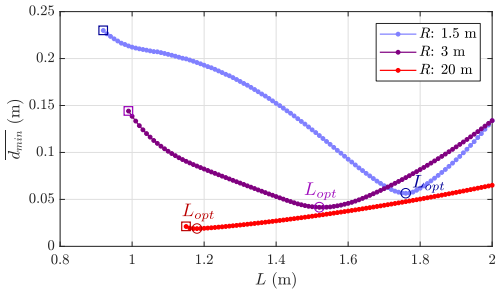

Fig. 4. MAE of d_{min} on a full lap of a circular path varying L. Constant speed: $1^m/_s$. Different radius: 1.5m, 3m and 20m.

Fig. 4 shows the results for three different path radius (1.5m, 3m and 20m) when parameter L is varied in each simulation by steps of 0.01m. Each point on the graph corresponds to a full lap simulation. The initial value of L on each of the three cases, denoted with a square on the plot, is the first one that does not make the system unstable. That is, smaller values of L make the vehicle's trajectory become unstable. Note that for each radius, there always exists a value of L that achieves the minimum error. In this paper, this value is represented by L_{opt} (optimal L).

B. Optimal L value in function of vehicle's velocity and path radius

The results showing how L_{opt} changes with the vehicle's reference velocity (V_{ref}) and the path radius (R) are presented in Fig. 5. The reference paths are circumferences again. Each point on the graph shows the value of L_{opt} for a given velocity reference and path radius. L_{opt} was obtained with an exhaustive discrete search (step of 0.01m). The behavior of L_{opt} in function of V_{ref} and R can be divided in three zones, represented by A, B and C in Fig. 5. Zone C shows constant L_{opt} with regard to the path radius. The speed of the vehicle here is slow enough to assume the path as a straight line. Thus, the value of L_{opt} corresponds to the value of the optimal L for a straight path. Zone B corresponds to the regular behavior of L_{opt} for a circular path. Finally, zone A is the one where the vehicle moves too fast for the given path radius and it is not able to follow the path correctly. There is no value of L that makes the vehicle converge to the path and follow it, due to the kinematic constrains of the plant.

IV. ADAPTIVE PARAMETER SELECTION

As seen in the previous section, the selection of parameter L depends on the velocity of the vehicle and the radius of the path. To develop an adaptive approach for the *NonLinear Guidance Law* it is necessary to define a function (or algorithm) that computes the value of L_{opt} given the vehicle's velocity and path radius. The output of this function has to be as similar as possible to the set of points obtained in Fig. 5.

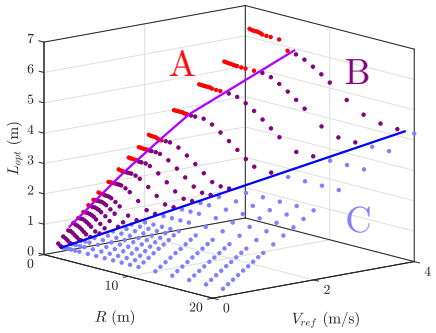

Fig. 5. L_{opt} in function of the path radius and vehicle's velocity.

In this paper a neural network (NN) is employed to fit a surface to the given set of points. This problem can also be solved using other types of approximations (polynomial, wavelets, etc.), however, since the shape of the surface to be fitted is similar to a sigmoidal function the authors considered the NN to be the best option. The details of this NN are explained below.

A. Neural network

The network architecture consists in three layers full connected in feedforward sequence. The inputs of the NN are the radius of the path and the velocity of the vehicle. The output is parameter L_{opt}. Hidden layers have 3, 6 and 3 neurons, respectively. The activation function of the neurons is sigmoidal, chosen mainly because of its resemblance with the surface required to fit. This architecture was found as the one that fits better the surface minimizing the overfitting problem. The training algorithm used is the Levenberg-Marquardt method with a regularization term. The regularization term is included to reduce the average generalization error, in other words, to avoid the overfitting problem as well. Fig. 6 shows the surface function obtained by the neural network with the training points denoted in red.

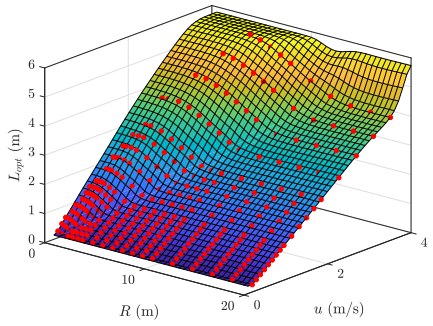

Fig. 6. Neural Network surface and its training points.

B. Obtaining the path radius

With the developed NN it is possible to obtain the optimal value of L, given the current x-body velocity of the vehicle

and the radius of the path. The radius of a continuous parametrized path defined as in Eq. (1) can be calculated with the inverse of the path curvature (k), given by

$$k(\gamma) = \frac{1}{R(\gamma)} = \frac{\left\| \frac{d\mathbf{p}_d(\gamma)}{d\gamma} \times \frac{d^2\mathbf{p}_d(\gamma)}{d\gamma} \right\|}{\left\| \frac{d\mathbf{p}_d(\gamma)}{d\gamma} \right\|^3} \quad (6)$$

The value of γ used to calculate k (or equivalently, R) must be chosen carefully. One could be tempted to use $\gamma_{d_{min}}$, but that is not a sensible choice. It would be equivalent, when driving a car, to steer it when you are already on the curve. Hence, it is necessary to have an anticipation distance to have the capability of adapting to the curves that come.

A new simulation scenario was designed to determine the optimal anticipation distance for each curve. It consists on a straight path of 10 meters long followed by a semicircle. The vehicle is required to follow the path at constant speed. In each simulation case, the reference velocity and the radius of the semicircle are changed. An anticipation distance, computed along the path, is used to obtain R, which is fed to the NN. The instantaneous value of parameter L, determined by the neural network, is used afterwards to obtain the VTP. From these simulations, an optimal anticipation distance for each case given by the vehicle's velocity and semicircle radius is obtained. The optimal anticipation distance is the one that achieves the best performance in terms of path distance error. The set of anticipation distances (${d_a}^*$) is approximated by a linear function of speed (u) and radius (R), Eq. (7).

$${d_a}^* = 1.5544u - 0.1787R + 0.2053 \quad (7)$$

Eq. (7) computes the optimal anticipation distance, which is needed to obtain R to feed the NN. Paradoxically, it is not possible to determine the distance to evaluate R without knowing R. Nevertheless, the maximum anticipation distance (d_A) is given when the radius is 0 and the minimum anticipation distance (d_a) is obtained with the maximum possible radius, noted R_{max}. This radius is apparent in Fig. 5 as for each u there is always a R_{max} in which, for all $R > R_{max}$, the L_{opt} is constant. The value of this radius can be obtained with the function defined in Eq. (8), which approximates the edge between B and C zones by a linear function. With this minimum and maximum distances an anticipation distance window is defined (Fig. 7).

$$R_{max} = 4.991u + 0.0333 \quad (8)$$

The anticipation window might contain several curves, but there is always one that is most urgent to consider. This curve determines R to be used as an input of the neural network. Curves are more urgent when they are closer to the vehicle and their radius is smaller. The criterion employed to obtain the radius of the most urgent curve, R_u, is stated as

$$R_u := \min_{\gamma} \left(R(\gamma) \mid {d_a}^* > d_\gamma \right) \quad (9)$$

where d_γ is the distance along the path between $\gamma_{d_{min}}$ and the evaluated γ inside the anticipation window. That is, R_u is the minimum possible radius on the window, $R(\gamma)$, which satisfies that the optimal anticipation distance, obtained with Eq. (7), is greater than d_γ.

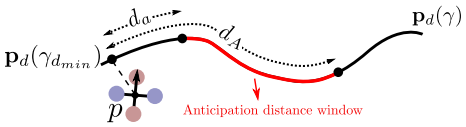

Fig. 7. Anticipation distance window to obtain the path radius (input of the NN).

C. Velocity reduction

To assure that the vehicle does not enter to zone A of Fig. 5, in which it is not able to follow the path, a maximum velocity reference has been defined, as shown in Eq. (10). This velocity depends on the path radius, and it is computed from the edge function between the A and B zones shown in Fig. 5, which can be approximated by an inverse tangential function. This expression has been obtained by means of the MATLAB curve fitting tool.

$$V_{ref_{max}} = 1.886 \arctan(0.6197R - 1.037) + 1.783 \quad (10)$$

V. STABILITY ANALYSIS

This sections gives conditions for the stability of the adaptive *NLGL*. In [5] the stability of the *NLGL* algorithm, implemented on a UGV, is analyzed and a stability condition for parameter L is given.

The UGV and the quadrotor present quite different dynamics. Nevertheless, *NLGL* is implemented using a SGC structure (i.e. with an autopilot). As the autopilot is in charge of controlling the inner dynamics of the vehicle, the *NLGL* algorithm only takes into account the kinematic model of the vehicle. If it is imposed that the quadrotor can only be driven on the x-body direction (no lateral movement), as it is already assured with the presented *NLGL* algorithm, it results that the kinematic models of the UGV and the quadrotor are very similar. The most relevant difference is that the UGV moves in the 2D space, while the quadrotor moves in 3D. However, as discussed in Sec. II-D the autopilot controls the altitude, therefore *NLGL* is only in charge of controlling the movement in the xy plane. In summary, both kinematic models can be considered equivalent. Thus, the results from [5] are taken to prove the stability of the proposed approach.

The stability condition derived in [5] is stated in Eq. (11), where u is the velocity of the vehicle in the x body axis, τ is the time constant of the yaw angle dynamics and $k_{\mathbf{p}_d}$ is the curvature (Eq. (6)) of a point on the path.

$$L > u\,\tau \sqrt{\frac{2}{1+k_{\mathbf{p}_d}^2} + \frac{2}{k_{\mathbf{p}_d}^2\left(1+k_{\mathbf{p}_d}^2\right)} + \frac{2}{k_{\mathbf{p}_d}^2\sqrt{1+k_{\mathbf{p}_d}^2}}} \quad (11)$$

Fig. 8 shows the limit of the stability condition of Eq. (11) (in purple) and the limit of stability presented in the simulation model (in red) in function of the radius for a vehicle's speed of $u = 1^m/_s$. As seen, they are quite similar, specially for large radius, being the stability condition more restrictive. Also, it is shown the optimal L value fitted by the NN (blue color), which fulfills the stability condition. It is important to

mention that this plot only corresponds to a vehicle's velocity of $1^m/_s$, however, it has been also verified that the NN output fulfills the stability condition for all its surface (velocity from $0.2^m/_s$ to $4^m/_s$, and radius from 0.5m to 20m).

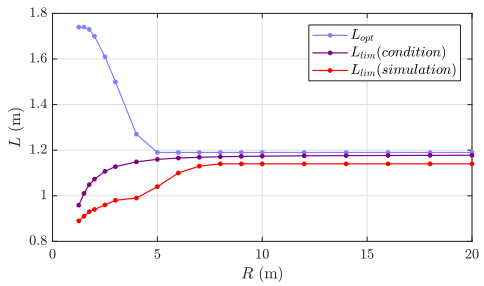

Fig. 8. Optimal L (blue), stability condition limit for L (purple) and stability limit in simulation (red) in function of the path radius (u = $1^m/_s$).

VI. RESULTS

Simulations results compare the performance of the proposed *Adaptive NLGL* algorithm with the regular *NLGL*. Two path references are analyzed: an eight-shaped 2D path and a spiral 3D path.

A. Eight-shaped path

The eight-shaped path is defined by Eq. (12). The altitude is fixed at 3m. And the amplitude is A = 2.5m.

$$\mathbf{p}_d(\gamma) = \begin{bmatrix} 2A\cos(\gamma) \\ A\sin(2\gamma) \\ 3 \end{bmatrix} \tag{12}$$

Fig. 9 shows the evolution on the xy plane of the UAV controlled by the *Adaptive NLGL* with the velocity reduction term (red) and the original *NLGL* (blue). The velocity reference is $1^m/_s$. The vehicle starts at $(x, y) = (5, 0)$ in hover conditions. Parameter L of the original *NLGL* was chosen minimizing path distance error for this particular path.

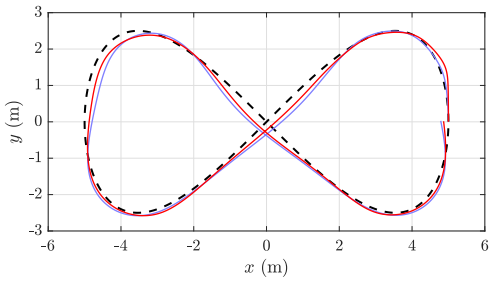

Fig. 9. Trajectory on the xy plane of *NLGL* (blue) and *Adaptive NLGL* with velocity reduction (red) (Eight-shape path, V_{ref} = $1^m/_s$).

Table I shows MAE of the path distance, the total time and the average velocity obtained by three variants of the *NLGL* algorithm performing one lap on the eight-shaped path. These variants are: the original *NLGL*, the proposed *Adaptive NLGL* and the *Adaptive NLGL* with the velocity reduction term. As shown in the results, the *Adaptive NLGL* with the velocity reduction is the one that presents the smallest distance error, but a slightly lower average velocity. It is important to highlight that the *Adaptive NLGL* has better performance than the regular *NLGL*.

TABLE I

RESULTS FOR ONE LAP OF THE EIGHT-SHAPED PATH ($V_{ref} = 1^m/_s$).

	$\overline{\mathbf{d}}$ **(m)**	**time (s)**	$\overline{\|\mathbf{v}\|}$ $(^m/_\mathbf{s})$
NLGL	0.1282	32.8375	0.9339
Adaptive NLGL	0.1054	32.9414	0.9338
Adaptive NLGL + Vel. red.	0.0947	33.4749	0.9146

Fig. 10 shows the evolution in time of parameter L computed by the *Adaptive NLGL* (green) and *Adaptive NLGL* with velocity reduction (red) when performing a full lap of the reference path. The velocity reduction term reduces the velocity in the curves which, at the same time, makes the algorithm reduce parameter L.

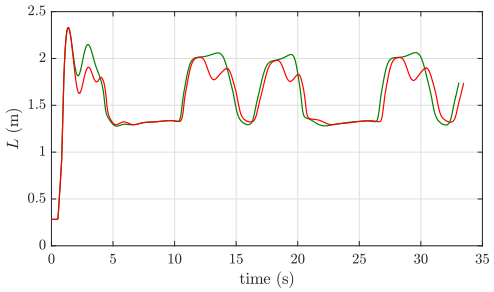

Fig. 10. Evolution of L with the *Adaptive NLGL* (green) and *Adaptive NLGL* with velocity reduction (red) (Eight-shaped path, V_{ref} = $1^m/_s$).

B. Spiral path

The spiral path is defined by Eq. (13).

$$\mathbf{p}_d(\gamma) = \begin{bmatrix} \gamma/2\cos(\gamma) \\ \gamma/2\sin(\gamma) \\ \gamma + 3 \end{bmatrix} \tag{13}$$

Fig. 11 shows the evolution in space of the regular *NLGL* (blue) and *Adaptive NLGL* with velocity reduction (red) following the spiral path with a velocity reference of $2^m/_s$. Again, the L of the original *NLGL* was tunned to minimize path distance error for the given path and velocity.

A comparison of the obtained results with the three variants of the *NLGL* algorithm when following the spiral path is found in Table II. This table shows the mean absolute error, the total time and the average velocity exhibited by the three variants when traveling from 0 rad to 6π rad of the spiral path. Again, the proposed *Adaptive NLGL* with the velocity reduction term provides the best performance and the simple *Adaptive NLGL* performs better than the *NLGL*.

VII. CONCLUSIONS

This paper analyses the effect of the scalar parameter L on the performance of the *NLGL* algorithm when it is applied to a quadrotor vehicle. The mean absolute path distance error

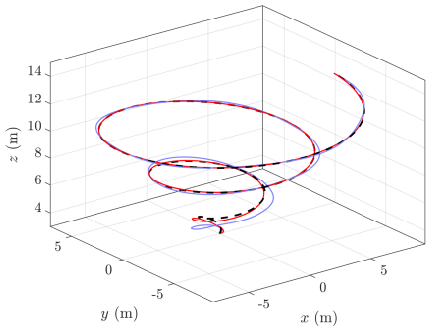

Fig. 11. Trajectory of *NLGL* (blue) and *Adaptive NLGL* with velocity reduction (red) (Spiral path, $V_{ref} = 2^{m}/_{s}$).

TABLE II

RESULTS FOR ONE LAP ON THE SPIRAL PATH ($V_{ref} = 2^{m}/_{s}$).

	$\overline{\mathbf{d}}$ **(m)**	**time (s)**	$\overline{\|\mathbf{v}\|}$ $(^{m}/_{\mathbf{s}})$
NLGL	0.2205	53.2537	1.7899
Adaptive NLGL	0.1787	52.2592	1.7866
Adaptive NLGL + Vel. red.	0.0730	55.3999	1.6563

is used as a performance indicator. The optimal value of L is analysed and it is shown to change depending on the vehicle's velocity and the path radius.

From these analysis, an adaptive law by means of a neural network is developed. This NN computes the optimal value of L from the vehicle's velocity and the value of the path radius. An algorithm has been developed to find the most restrictive radius on the path on a defined anticipation distance window. The anticipation distance window depends on the vehicle's velocity. Then, a velocity reduction term has been added to the *Adaptive NLGL* to decrease the speed when the radius of a curve is too small to be followed by the vehicle at the current velocity.

Simulation results compare the performance of the regular *NLGL*, the proposed *Adaptive NLGL* and the *Adaptive NLGL* with the velocity reduction term with two different reference paths: an eight-shaped 2D path and a spiral 3D path. The results show that the *Adaptive NLGL* achieves better performance than the *NLGL*. Also, adding the velocity reduction term, makes the vehicle follow with higher accuracy the smaller radius curves. Furthermore, it is important to mention that the main advantage of the *Adaptive NLGL* is that, as opposed to the standard *NLGL*, it is not necessary to tune its parameters when the reference path is changed.

Current work is focused on the study of the behaviour of L_{opt} in function of the parameter τ, aiming to develop a general Adaptive *NLGL* that can be used on other UAVs with different inner dynamics.

ACKNOWLEDGMENT

This work has been funded by: FEDER/Ministerio de Ciencia, Innovación y Universidades - Agencia Estatal de Investigación (project ref. DPI2017-88403-R). Bartomeu Rubí is also supported by the Secretaria d'Universitats i Recerca de la Generalitat de Catalunya, the European Social Fund (ESF) and AGAUR under a FI grant (ref. 2017FI_B_00212).

REFERENCES

[1] D. Cabecinhas, R. Cunha, and C. Silvestre, "Rotorcraft path following control for extended flight envelope coverage," in *Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 28th Chinese Control Conference (CDC/CCC)*, 2009, pp. 3460–3465.

[2] S. Su, "Path following control of non-minimum phase VTOL aircraft via minimum distance projection method," in *26TH Chinese Control and Decision Conference (CDC)*, 2014, pp. 708–712.

[3] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotovic, "Performance limitations in reference tracking and path following for nonlinear systems," *Automatica*, vol. 44, no. 3, pp. 598–610, 2008.

[4] A. Micaelli, C. Samson, P. Robotique, and P. Icare, "Trajectory tracking for unicycle-type and two-steering-wheels mobile robots," in *IFAC Proceedings Volumes*, 1994, pp. 249–256 Vol.27.

[5] G. Heredia and A. Ollero, "Stability of autonomous vehicle path tracking with pure delays in the control loop," *Advanced Robotics*, vol. 21, no. 1-2, pp. 23–50, 2007.

[6] P. B. Sujit, S. Saripalli, and J. B. Sousa, "Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless," *IEEE Control Systems*, vol. 34, no. 1, pp. 42–59, 2014.

[7] A. Ratnoo, S. Hayoun, A. Granot, and T. Shima, "Path following using trajectory shaping guidance," *Journal of Guidance, Control and Dynamics*, vol. 38, pp. 106–116, 2015.

[8] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for miniature air vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 519–529, 2007.

[9] A. Gautam, P. B. Sujit, and S. Saripalli, "Application of guidance laws to quadrotor landing," in *International Conference on Unmanned Aircfraft Systems (ICUAS)*, 2015, pp. 372–379.

[10] G. V. Pelizer, N. B. F. Silva, and K. R. L. J. C. Branco, "3D path-following algorithms for unmanned aerial vehicles adjusted with genetic algorithm," in *Communication in Critical Embedded Systems*, 2017, pp. 63–80.

[11] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[12] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2004, pp. 4393–4398 Vol.5.

[13] Y. Chen, J. Yu, Y. Mei, Y. Wang, and X. Su, "Modified central force optimization (MCFO) algorithm for 3D UAV path planning," *Neurocomputing*, vol. 171, pp. 878–888, 2016.

[14] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[15] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, E. De Lellis, and A. Pironti, "Path generation and tracking in 3-D for UAVs," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 4, pp. 980–988, 2009.

[16] T. Yamasaki, S. N. Balakrishnan, and H. Takano, "Integrated guidance and autopilot for a path-following UAV via high-order sliding modes," in *2012 American Control Conference (ACC)*, 2012, pp. 143–148.

[17] N. Dadkhah and B. Mettler, "Control system design and evaluation for robust autonomous rotorcraft guidance," *Control Engineering Practice*, vol. 21, no. 11, pp. 1488–1506, 2013.

[18] S. Park, J. Deyst, and J. How, "Performance and lyapunov stability of a nonlinear path-following guidance method," *Journal of Guidance, Control and Dynamics*, vol. 30, pp. 1718–1728, 2007.

[19] N. Cho, Y. Kim, and S. Park, "Three-dimensional nonlinear differential geometric path-following guidance law," *Journal of Guidance Control and Dynamics*, vol. 38, no. 12, pp. 2366–2385, 2015.

[20] B. Rubí, A. Ruiz, R. Pérez, and B. Morcego, "Path-flyer: A benchmark of quadrotor path following algorithms," in *2019 15th IEEE International Conference on Control and Automation*, 2019.