A Comparative Analysis and Design of Controllers for Autonomous Bicycles

Niklas Persson, Tom Andersson, Anas Fattouh, Martin C. Ekström, Alessandro V. Papadopoulos

***Abstract*— In this paper, we develop and compare the performance of different controllers for balancing an autonomous bicycle. The evaluation is carried out both in simulation, using two different models, and experimentally, on a bicycle instrumented with only lightweight components, and leaving the bicycle structure practically unchanged. Two PID controllers, a Linear Quadratic Regulator (LQR), and a fuzzy controller are developed and evaluated in simulations where both noise and disturbances are induced in the models. The simulation shows that the LQR controller has the best performance in the simulation scenarios. Experimental results, on the other hand, show that the PID controllers provide better performance when balancing the instrumented bicycle.**

I. Introduction

Modern vehicles, equipped with sensors for mapping the surrounding environment and to detect and classify other road users struggle when it comes to detection of bicycles [1]. When the vehicle's autonomous emergency braking system is tested by the car safety performance assessment program, EuroNCAP, a bicycle with a dummy on top is placed on a moving platform[1]. The platform then moves in a straight line in front or beside the vehicle being tested. A riderless bicycle which is design to have a minimal impact on its resemblance and can manoeuvre realistically would improve the testing environment for autonomous vehicles.

The control of a bicycle motion is an interesting research problem, that has been investigated for decades [2], with several different variants [3]. Its configuration with two inline wheels makes the bicycle a statically unstable system, making the control of the bicycle dynamics a very interesting control problem at low speeds [3]. A cyclist uses a combination of regulation on the forward speed, the steering angle, and the lean angle to balance the bicycle. A similar approach can also be used to control a driverless bicycle [3], [4]. However, direct control of the lean angle requires a flywheel, an inverted pendulum or something similar to be mounted on a bicycle and will, therefore, alter the appearance of the bicycle to a large degree, as well its usability.

Different control approaches have been proposed in the literature to design an autonomous bicycle, ranging from model-free to model-based [5], [2], from control-theoretic to machine learning [6], [7]. Several such approaches are evaluated in simulation, and the results strictly depend on the level of accuracy of the bicycle model. Furthermore, aspects like the computational complexity, the execution time, and implementation issues are hardly discussed.

As a first step to develop a riderless bicycle which can be used for testing of autonomous vehicles safety systems, several control strategies which are commonly used in literature for controlling bicycles are investigated and compared in this paper. The controllers are evaluated both in simulation, and on an instrumented bicycle running on a bicycle roller, to understand what is the more effective approach. Their performance is assessed both in terms of the ability of balancing the bicycle, and in terms of their execution time on the embedded control platform. In particular, three main control strategies are designed and compared: (i) a Proportional Integral Derivative (PID) controller, (ii) a Linear Quadratic Regulator (LQR), and (iii) a fuzzy controller.

II. Related Work

In the past, several researchers focused on the design of suitable bicycle dynamic models. The *Whipple model* [8] is often utilised such as in the work by Baquero-Suárez *et al.* [6] and Mejiard *et al.* [9]. The main drawback of the Whipple model is that it uses steering and lean torque as control variables which may be difficult to realise on a real autonomous bicycle. Moreover, the instrumented bicycle used in this paper has no direct regulation of the lean angle. Another commonly used bicycle model is the *point-mass model*, also known as the *inverted pendulum model*. For example, Sharma *et al.* [10] utilised the point-mass model to produce a fuzzy controller for stabilising a bicycle. Hauser *et al.* [11] used the point-mass model to investigate trajectory tracking for a motorcycle. The point-mass model only has angular inputs and outputs, thus eliminating the problems of dealing with torques. Both the point-mass model and the Whipple model can be used for the control design, thanks to their simplicity. However, more accurate models are needed to validate the control design in simulation before going to the implementation on the real bicycle. In this paper, we model the bicycle using Adams[2], a multibody dynamics software, and a linear model based on the point mass model.

Numerous control structures have been proposed to balance an autonomous bicycle. Tan *et al.* [5] developed a reinforcement learning approach to teach a humanoid to

This work is part of a research project between Mälardalen University, Chalmers University, Volvo Cars, AstaZero, and Cycleurope and it is funded by Vinnova. The work is partly funded by Eskilstuna kommun och Eskilstuna Fabriksförening.

The authors are with the academy of Innovation, Design and Technique, Mälardalens University, Västerås, Sweden e-mail: Firstname.Surname@mdh.se.

[1] https://www.euroncap.com/en/vehicle-safety/the-ratings-explained/vulnerable-road-user-vru-protection/aeb-cyclist/

[2] https://www.mscsoftware.com/it/product/adams

balance a bicycle in simulation. Shafiekhani *et al.* [7] developed adaptive critic-based neuro-fuzzy controller to solve the same problem. However, Meehan and Ruina [12] highlights that the complexity of designing complex nonlinear controllers for balancing a bicycle is often not worth the small performance benefits with respect to simpler control strategies. In [12], an LQR is compared with an dynamic programming optimal controller and the results shows that the two controllers have almost identical basin of attraction under reasonable constrained steer angles and rates.

In the work of García *et al.* [13], an autonomous bicycle is modelled and evaluated in several different scenarios, including starting from a stationary conditions. Thanks to an innovative design for a flywheel mounted on the bicycle together with steering control, the bicycle managed to balance in a forward speed range between 0–6m/s. To control the flywheel, an LQR controller is designed and to control the steering torque an intuitive controller [14] is used. Though, the 7.5kg flywheel peaks at around 500rpm which would consume a lot of energy. Furthermore, according to He *et al.* [15], methods involving direct regulation of the lean angle usually struggle when it comes to the balance of regular size bicycles, where the weight and velocity are often increased compared to a small bicycle.

Alternatively, regulation of the steering angle can be used to stabilise the bicycle. Such an approach was used in the work of Tanaka and Murakami [16], where the lean angle and lean rate were used in a PD controller to compute the desired steering acceleration. Vatanshevanopakorn and Parnichkun [17] proposed an LQR optimised for a bicycle model coupled with the dynamics of an electrical steering motor. The simulation results presented show that the proposed LQR controller structure was capable of stabilising the bicycle with an initial lean angle and steering angle other than zero. An LQR was also utilised in the work of Anjumol and Jisha to control a second-degree bicycle model [18]. The results obtained was compared with a posture controlled proposed by Tanaka *et al.* [19], and concluded that the LQR performed better than the posture controller in simulation.

In [15], three proportional gains are used in a feedforward and feedback loop scheme to stabilise a bicycle in both simulation and experiments. The control scheme utilises measurements of both the lean angle and the lean angle rate to compute a desired steering angle of the handlebar. The bicycle used in experiments is equipped with two motors, a few sensors, a battery, and a compactRIO which serves as the main processing unit. The results of the experiments are impressive, however, the bicycle is quite massive with a weight of 52.5kg.

In this work, we conduct a comparative performance evaluation of the main control approaches, designed for the instrumented bicycle. The way the handlebar is controlled depends on the motor and motor controller used, it can either be by steering position [15], steering torque [6], or steering velocity [12]. We control the steering position when using the PID and the fuzzy controller and steering velocity for the LQR to understand what can be more beneficial for future autonomous bicycles. The bicycle is designed with lightweight components and without altering the main structure of a regular bicycle.

III. Modeling of Instrumented Bicycle

A. Experimental platform

The instrumented bicycle, illustrated in Fig. 1 is based on a regular-sized electrical bicycle of a male model with the propulsion motor in the rear wheel (#1). The 11.6Ah and 36V battery is mounted on the frame of the bicycle (#5). In the design process of the instrumented bicycle, care has been taken into both the size and the weight of the components to fit all extra components, such as IMU and main processing unit, in a bicycle basket in the upcoming iteration of the instrumented bicycle. A DCX32L Maxon motor together with a gear hub and encoder are utilised to control the handlebar through two cogwheels with a rubber band in between (#4). To control the steering motor a Junus motor controller is mounted on the side of the battery (#3) where the steering velocity ($\dot{\delta}$), is the input signal. Power distribution boards and the main processing unit, a National Instruments roboRIO (#2), are attached to the centre of the bicycle however on the opposite side of the one visualised in Fig 1. To measure the speed of the bicycle, a Hall sensor is used which measures the time between pulses of 12 evenly distributed magnets around the rear wheel (#1). To sense the lean angle a VectorNav VN-100 IMU is used and configured to output the lean angle and the lean rate of the bicycle (#6).

The roboRIO is equipped with both a dual-core ARM Cortex-A9CPU and an Artix-7 FPGA and the code is written using LabVIEW[3]. The FPGA is used for acquiring sensor data and actuating the motors. To actuate the steering motor using a steering position, a PD controller is implemented on the FPGA which takes the error between the current steering position δ and the desired one δ^* and computes a steering velocity fed to the bicycle $\dot{\delta}$ as presented in Fig. 2. The parallel structured PD controller is tuned experimentally and is executing at 600Hz with $K_P = 0.1$, $K_D = 0.04$, and a filter time constant $T_f = 0.8$. The different balancing controllers are realised on the CPU and execute at a frequency

[3]https://www.ni.com/sv-se/shop/labview.html

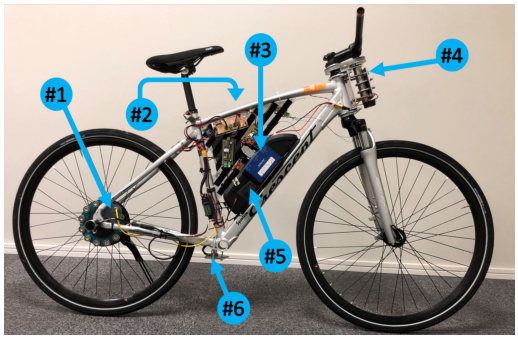

Fig. 1. Instrumented bicycle. (#1) Propulsion motor; (#2) NI roboRIO; (#3) Motor controller; (#4) Steering motor; (#5) Battery; (#6) IMU.

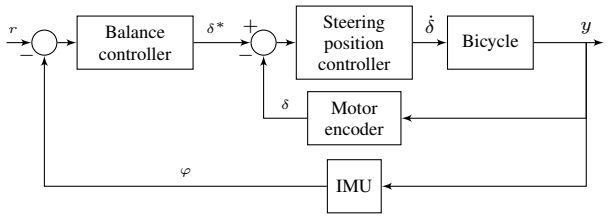

Fig. 2. Control structure of the instrumented bicycle where a PD controller is used in an inner loop to control the steering position.

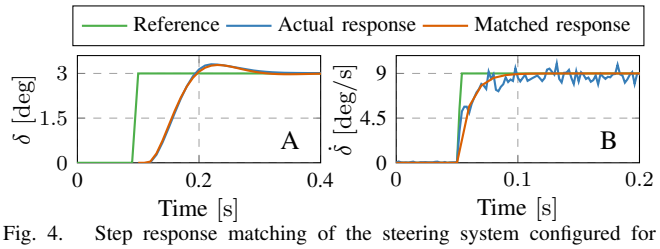

Fig. 4. Step response matching of the steering system configured for steering position (A), and steering velocity (B). The matched responses are used in simulation to model the steering system including the motor and friction.

of 100Hz. The main geometrical features of the instrumented bicycle are illustrated in Fig. 3 and given in Table I along with constraints on the lean angle, lean rate, steering angle, and steering rate.

TABLE I

PARAMETERS OF THE INSTRUMENTED BICYCLE.

Design parameters			
Parameter	**Symbol**	**Unit**	**Value**
CoG with respect to O (x)	a	[m]	0.473
CoG with respect to O (z)	h	[m]	0.515
Gravity	g	[m/s^2]	9.820
Wheelbase	b	[m]	1.080
Mass	m	[kg]	23.720
Wheel radius	r	[m]	0.349
Trail	c	[m]	0.087
Head angle	λ	[deg]	72.950
Constraints			
Lean angle	φ	[deg]	± 2
Lean rate	$\dot{\varphi}$	[deg/s]	50
Steer angle	δ	[deg]	± 15
Steer rate	$\dot{\delta}$	[deg/s]	70

B. Linear model

The point-mass model [3] describe the dynamics of the lean angle φ based on the steering angle δ and velocity $\dot{\delta}$ and is linearised about the equilibrium of zero lean and steer angle for Getz bicycle model [2]:

$$\ddot{\varphi} - mgh\varphi = \frac{v}{b}\dot{\delta} + \frac{mv^2 h}{b}\delta, \tag{1}$$

where v is the forward velocity of the bicycle, and the physical parameters are reported in Table I. The model assumes zero head angle ($\lambda = 0$), constant velocity (v), massless wheels and front fork. Instead, the total mass of the bicycle is lumped together forming a point mass m. The transfer function from δ to φ is

$$G(s) = \frac{\Phi(s)}{\Delta(s)} = \frac{av}{bh}\frac{s + \frac{v}{a}}{s^2 - \frac{g}{h}}. \tag{2}$$

However, as $G(s)$ does not model friction or dynamics of the steering setup including the steering position controller, $G(s)$ is put in series with another transfer function, $H(s)$. To obtain $H(s)$, a step response is recorded from the instrumented bicycle, see Fig. 4.A, and matched with a Second-Order Plus Dead Time (SOPDT) transfer function

$$H(s) = \frac{\Delta(s)}{\Delta^*(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} e^{-ds}, \tag{3}$$

where $\zeta = 0.6$, $\omega_n = 33.9$, and the time delay, $d = 0.015$. During the step response, the bicycle is held in an upright position with both wheels on the ground and the handlebar pointing forward. The input to $H(s)$ is the desired steering angle, δ^* and the output of $H(s)$ is the actual steering angle δ. The complete model, from δ^* to φ, is obtained as $P(s) = H(s)G(s)$ and is discretize using the zero-order hold method and a sample time $T_s = 0.01$ seconds.

For control strategies, such as LQR, the point-mass model given in (1) is utilised in its state-space representation, with:

$$\mathbf{A_1}(v) = \begin{bmatrix} 0 & 1 & 0 \\ \frac{g}{h} & 0 & -\frac{v^2}{bh} \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B_1}(v) = \begin{bmatrix} 0 \\ -\frac{av}{bh} \\ 1 \end{bmatrix}, \quad \mathbf{C_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{D_1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{4}$$

where, the state vector consists of $\mathbf{x}_1 = [\varphi, \dot{\varphi}, \delta]^\top$. Note that the input, $u_1 = \dot{\delta}$ represents the steering velocity, instead of the desired steering position which is the input signal to the model given by the transfer function $P(s)$. When the LQR is implemented on the instrumented bicycle, the PD controller for steering position is bypassed, and the steering velocity is fed directly to the motor controller.

To include the dynamics of the steering system, comparable to the approach of the transfer function $P(s)$, another step response is matched. However, the reference is a steering velocity instead of a steering position since the model in (4) is using the steering velocity as its input. As a reference angular velocity, 9deg/s is commanded to the DC motor mounted on the instrumented bicycle. The Junus motor controller, used for controlling the steering system, logs the angular velocity data for post-processing. The result is presented in Fig. 4.B.

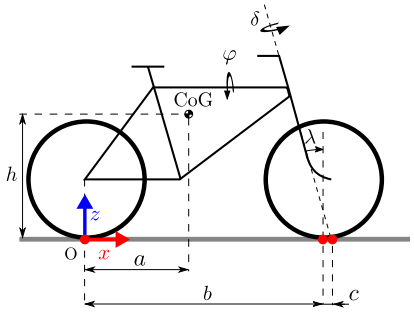

Fig. 3. Geometrical features of a bicycle, where a and h corresponds to the measures horizontal and vertical measure of the CoG. The wheelbase is denoted by b.

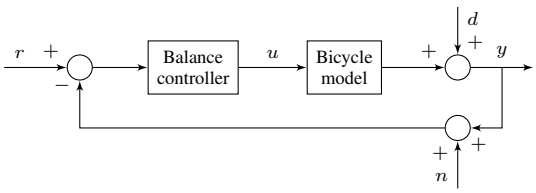

Fig. 5. Reference feedback control scheme for balancing a bicycle.

The matched step response in Fig. 4.B is converted to its state-space, and its matrices are:

$$\mathbf{A}_2 = \begin{bmatrix}-100\end{bmatrix}, \mathbf{B}_2 = \begin{bmatrix}1\end{bmatrix}, \mathbf{C}_2 = \begin{bmatrix}100\end{bmatrix}, \mathbf{D}_2 = \begin{bmatrix}0\end{bmatrix}, \quad (5)$$

with the state vector $\mathbf{x}_2 = \dot{\delta}$, and the control input $u_2 = \dot{\delta}^*$ representing the desired steering velocity. The two state spaces in 4 and 5 are combined in series as:

$$\mathbf{A}(v) = \begin{bmatrix}\mathbf{A}_2 & \mathbf{0} \\ \mathbf{B}_1\mathbf{C}_2 & \mathbf{A}_1\end{bmatrix}, \quad \mathbf{B}(v) = \begin{bmatrix}\mathbf{B}_2 \\ \mathbf{B}_1\mathbf{D}_2\end{bmatrix}, \\ \mathbf{C} = \begin{bmatrix}\mathbf{D}_1\mathbf{C}_2 & \mathbf{C}_1\end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix}\mathbf{D}_1\mathbf{D}_2\end{bmatrix}. \quad (6)$$

The input signal is the desired steering velocity $u = u_2 = \dot{\delta}^*$.

C. Nonlinear model

The instrumented bicycle was disassembled and each part was measured, weighed, and designed in SolidWorks[4]. Both the rear wheel motor and the steering motor are defined as rotary motors in SolidWorks. The complete model is then exported to Adams which is a multibody dynamics simulation software that uses the Newton-Euler method to obtain the equations of motion.

In Adams, the steering motor of the bicycle is defined as either steering position or steering velocity depending on the control structure used. Gravity and a contact surface between the wheels and the ground are included as well. To model the Columb friction force between the ground and the wheels, the static and dynamic friction coefficients $\mu_s = 0.7$ and $\mu_d = 0.7$ are used as well as a stiction transition velocity of 0.2m/s and a friction transition velocity of 1m/s.

IV. Control strategies

The closed-loop system in Fig. 5 is used to design control strategies for balancing the bicycle. The control signal u depends on which model is used as explained in the previous section. The reference signal is denoted with r, and the controlled variable with y. A load disturbance d, which intend to emulate a small physical side push of the bicycle is included in simulations. Additionally, the measurement noise n in the lean angle measurements is also considered.

A. PID controllers

Two PID controllers, tuned using different methods, are used in this paper. Both PID controllers are designed using the ideal form of a PID controller in discrete time

$$U(z) = K_P\left(1 + K_I \cdot T_s\frac{1}{z-1} + K_D \cdot \frac{1}{T_s}\frac{z-1}{z}\right), \quad (7)$$

[4]https://www.solidworks.com/

TABLE II
The gains for the two PID controllers

PID gains		
Gain	**LSPID**	**ATPID**
K_P	2.514	3.167
K_I	1.544	1.326
K_D	0.074	0.069

with $T_s = 0.01$ seconds. The first PID controller – in the following referred as LSPID – is tuned using a loop shaping method explained in detail in the work by Andersson *et al.* [20]. The general objectives of the loop shaping method is to ensure good stability, tracking performance and robustness to disturbances and uncertainties [21]. The control loop objectives are formulated as the following constraints on the loop transfer function frequency response:

- The target bandwidth is selected so that the open loop system should cross the 0 dB mark once with a phase margin of at least 30°
- For disturbance rejection, the gain for the frequency response of the open loop system below the target bandwidth should be high
- To assure robustness of plant uncertainties, the gain for the frequency response of the open loop system above the target bandwidth should be low

The cost function J, is defined as:

$$\begin{aligned} J =& w_1(\omega_b - \omega_t)^2 \\ &+ w_2\int_{\omega_b}^{\pi/T_s} 20\log|K(j\omega)P(j\omega)|\,\mathrm{d}\omega \\ &- w_3\int_{0}^{\omega_b} 20\log|K(j\omega)P(j\omega)|\,\mathrm{d}\omega, \end{aligned} \quad (8)$$

where $\omega_t = 15$rad/s is the target bandwidth and ω_b is the bandwidth of the loop transfer function defined as:

$$\omega_b = \inf_{\omega}|K(j\omega)P(j\omega)| \leq 1, \quad (9)$$

and weights are chosen as $w_1 = 0.05$, $w_2 = w_3 = 5\times10^{-4}$.

An alternative tuning of the PID controller, referred to as ATPID, is performed using the Simulink PID tuner applied on the linear model $P(s)$. A cross-over frequency $\omega_c = 15$rad/s and phase margin $\phi_m = 60^\circ$ were chosen in the design process which results in a stable controller with a rise time of 0.05s and settling time of 1.44s. The gains for the respective PID controller are reported in Table II.

B. Linear quadratic regulator

The LQR [22] in discrete time, is set to minimise the cost function:

$$J(\mathbf{u}) = \sum_{k=1}^{\infty}(\mathbf{x}(k)^\top\mathbf{Q}\mathbf{x}(k) + \mathbf{u}(k)^\top\mathbf{R}\mathbf{u}(k)) \quad (10)$$

where the $\mathbf{Q}$ and $\mathbf{R}$ are semi-definite positive matrices. The state-feedback control law is given by:

$$\mathbf{u}(k) = -\mathbf{K}\mathbf{x}(k-1) \quad (11)$$

where $\mathbf{K}$ is computed as

$$\mathbf{K} = (\mathbf{R} + \mathbf{B}^\top\mathbf{P}\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{P}\mathbf{A} \quad (12)$$

TABLE III

THE FUZZY RULE SET WITH *"And"* LINGUISTIC INTERCEPTION TERM.

$\mathbf{e}_\varphi$ / $\mathbf{\Delta e}_\varphi$	**NL**	**NM**	**NS**	**Z**	**PS**	**PM**	**PL**
NL	NL	NL	NM	NM	NS	NS	Z
NM	NL	NM	NM	NS	NS	Z	PS
NS	NM	NM	NS	NS	Z	PS	PS
Z	NM	NS	NS	Z	PS	PS	PM
PS	NS	NS	Z	PS	PS	PM	PM
PM	NS	Z	PS	PS	PM	PM	PL
PL	Z	PS	PS	PM	PM	PL	PL

and $\mathbf{P}$ is obtained by solving the discrete-time algebraic Riccati equation:

$$\mathbf{P}(k-1) = -\mathbf{A}^\top\mathbf{P}(k)\mathbf{B}(\mathbf{B}^\top\mathbf{P}(k)\mathbf{B}+\mathbf{R})^{-1}\mathbf{B}^\top\mathbf{P}(k)\mathbf{A} + \mathbf{Q} + \mathbf{A}^\top\mathbf{P}(k)\mathbf{A}. \quad (13)$$

The matrices $\mathbf{A}$ and $\mathbf{B}$, the state vector $\mathbf{x}$, and the input $\mathbf{u}$ are given by the state-space model of the system (6), converted to discrete time with a sampling time $Ts = 0.01$s. The $\mathbf{Q} = \text{diag}(Q_{ii})$ and $\mathbf{R} = \text{diag}(Q_{jj})$ matrices were chosen according to Bryson's Rule [23], as

$$Q_{ii} = \frac{1}{\text{Max value of } x_i^2} \quad R_{ii} = \frac{1}{\text{Max value of } u_i^2} \quad (14)$$

with the constraints found in table I. Now, by utilising equation (12) the following state-feedback gain, $\mathbf{K}$, is obtained:

$$\mathbf{K} = \begin{bmatrix} 22.46 & -37.35 & -4.91 & 8.77 \end{bmatrix}^\top. \quad (15)$$

On the instrumented bicycle it is possible to access all states except the steering velocity, which is estimated from the steering position as:

$$\dot{\delta}(t) \approx \frac{\delta(t) - \delta(t-T_s)}{T_s}, \quad T_s = 0.01\text{s}. \quad (16)$$

C. Fuzzy controller

The fuzzy controller is inspired by the work of Abdolmalaki [24] and modified using a trial and error approach. The controller uses two inputs, the lean angle and the lean angle difference, and computes the desired steering angle, hence the linear model $P(s)$ is utilised. The input and output membership functions are shown in Fig. 6 where NL, NM, NS, Z, PS, PM, PL are short for Negative Large, Negative Medium, Negative Small, Zero, Positive Small, Positive Medium, and Positive Large respectively. The fuzzy rule set is defined in Table III, using the same abbreviations as in Fig. 6. The columns represent the lean error e_φ and the rows represent the lean error difference Δe_φ. To represent implication and the "And" operation, the product function is utilised. The methods of aggregation and defuzzification are represented by the sum and the centroid functions respectively.

V. RESULTS

To evaluate the performance of the different controllers in simulation the Integrated Squared Error (ISE) of the lean angle and rejection of lean angle disturbances is used. For the experimental results, the execution time of the controllers on the platform is considered as well as their maximum balancing time of the bicycle on a bicycle roller.

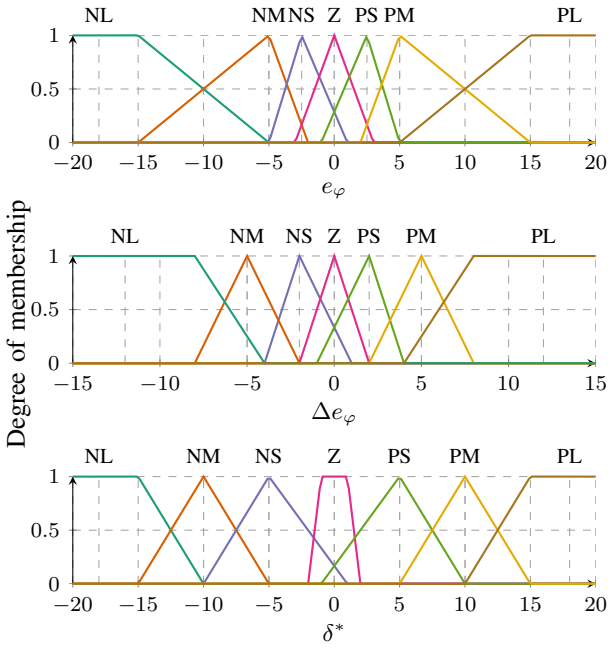

Fig. 6. The fuzzy controllers membership functions, the first membership function is the lean angle error e_φ and the second membership function is the lean angle error difference Δe_φ. The output membership function presented in the bottom graph correlates to the desired steering angle δ^*.

A. Simulation setup

To evaluate the behaviour of the different controllers and models in simulation, Mathworks Simulink[5] is utilised. The nonlinear model is exported from ADAMS and used in co-simulation through Simulink where it is put in series with the transfer functions of the step response matching to capture the dynamics of the steering setup. Both the linear and nonlinear models are set up in continuous time and the controllers are using a sampling time $T_s = 0.01$s, to transfer the data between the time domains rate transition blocks are utilised. The forward speed during simulations on both the linear and nonlinear model is set to 14km/h and reference lean r of 0 degrees. In simulations, Gaussian noise is added to the lean angle measurements with a standard deviation of $0.01°$. The noise is measured by placing the VN-100 on a flat surface and collect the roll angle data for 30 minutes, this is repeated 3 times. To simulate a gentle push on the bicycle, a disturbance is induced in the lean angle measurements with an amplitude of 1 degree and lasts for 0.25s. The disturbance is activated after 5s. The constraints of the steering angle and steering velocity presented in Table I are implemented in simulation as saturation of the control signals.

B. Simulation results

The lean and steering angle of both the nonlinear and the linear models are shown in Fig. 7. The grey area in the subplots indicates where the disturbance in the lean angle measurements is injected. In the nonlinear case, the bicycle starts from zero and instantly accelerates up to 14km/h which

[5] https://se.mathworks.com/products/simulink.html

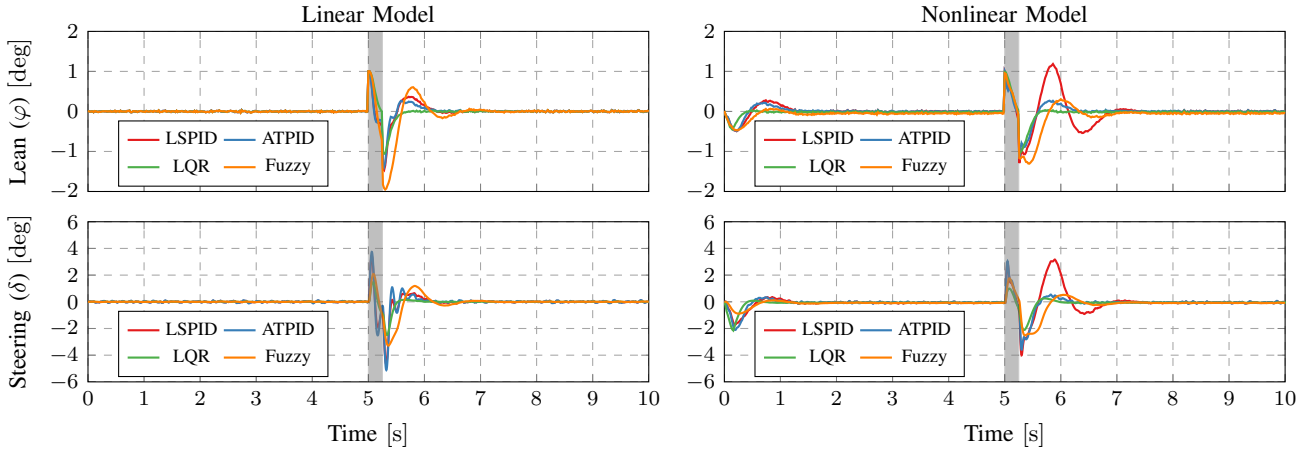

Fig. 7. Results of the different control strategies when controlling the linear model (left) and the nonlinear model (right) in simulation.

TABLE IV

THE ISE VALUES FROM THE LEAN ANGLE ERROR IN SIMULATION

Integrated Squared Error (ISE) on lean angle		
	Nonlinear model	**Linear model**
LSPID	83.79	30.25
ATPID	31.25	24.16
LQR	27.24	23.39
Fuzzy	71.47	88.55

TABLE V

EXECUTION TIME OF THE DIFFERENT CONTROL STRATEGIES

Execution time		
Controller	**Mean [μs]**	**Standard deviation [μs]**
PID	8.24	4.78
LQR	9.97	3.96
Fuzzy	582.37	54.50

makes the lean angle, and in extension the steering angle, to deviate a bit from zero in the beginning. The Integrated Square Error (ISE) are computed for the lean angle of the bicycle and presented in Table IV.

The most consistent control structure in simulation is the LQR which shows promising results for both models, for the linear case, this is not surprising as the LQR is optimised for that model. As the LQR behaves similarly on both models this suggests that the linear model is a good approximation of the more complex nonlinear model. However, it is obvious from the results of the PID controllers that the nonlinear model is more challenging to control for this type of controller. Especially the LSPID which has smaller gains struggles when it comes to disturbance rejection of the nonlinear model, this is confirmed by the ISE values for the LSPID. The ATPID, with a set of higher gains, are performing much better on both models. The ISE also verifies the consistency of the LQR and shows that the Fuzzy controller struggles on both models.

C. Experimental setup

In experiments, the instrumented bicycle is placed on a bicycle roller and reference lean of 0 degrees is used for all controllers[6]. The reason for choosing a bicycle roller is due to space limitations and weather conditions. At startup, the handlebar is pointing approximately forward and the bicycle is held by an operator in an upright position. After the bicycle has accelerated up to 14km/h, the operator releases the bicycle and the logging of data begins. In case the operator touches the bicycle, the remaining data does not qualify as self stabilising of the bicycle, which is indicated by the grey areas in Fig. 8. There were no disturbance injected in the lean angle measurements in the experiments, mainly due to the experimental setup using a bicycle roller.

[6]A footage of one experiment is available at this link: `https://youtu.be/9owSiGU-Z0o`

D. Experimental results

The outcome of the experiments conducted on the roller is presented in Fig. 8. All four controllers manage to balance the bicycle. However, due to the narrow bicycle roller, the instrumented bicycle tends to go out of bounds and an operator needs to assist the bicycle to keep it on the roller. The Fuzzy controller manages to balance the bicycle, but with large oscillations in the lean angle which makes it drift off the roller after 10 seconds. Perhaps with a better tuned fuzzy system, the bicycle balancing could be improved. However, the long execution time of the fuzzy system makes it a weak candidate compared to LQR and PID on a real-time embedded system. The two PIDs and the LQR manages to stabilise and keep the bicycle on the roller for over 40 seconds, with the LQR performing slightly worse than the PID's. A reason for this might be caused by the estimation of the steering velocity, which could be improved by a Kalman filter. The results also suggest that in experiments, it is more beneficial to use the steering position as the control signal compared to the steering velocity.

The LSPID, which did not show the most promising results in simulation, produced the best results in experiments, indicating that the nonlinear validation model could be improved in future work. The most consistent controllers, when considering both experiment and simulation, are the ATPID and LQR.

To evaluate the execution time of the different control structures they are implemented on the roboRIO platform. Random numbers in the range $[-15, 15]$deg are used as inputs to the controllers and the mean execution time and

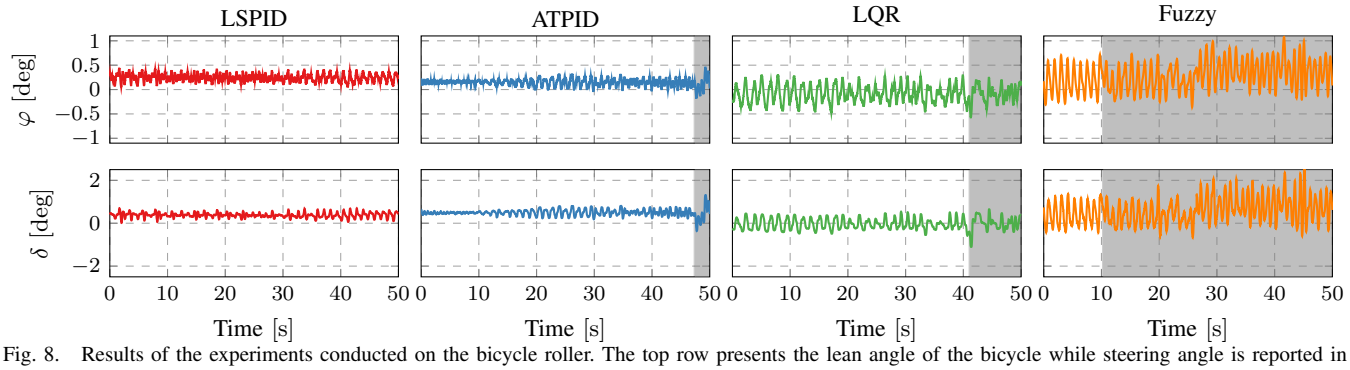

Fig. 8. Results of the experiments conducted on the bicycle roller. The top row presents the lean angle of the bicycle while steering angle is reported in the bottom row. The gray area indicates that the bicycle has been aided by an operator, thus the experiment is considered to be aborted at that point.

standard deviation of 10 000 loops are calculated for each controller. Table V shows the obtained averages for the three classes of controllers. Both the PID and the LQR show an execution time of few micro-seconds, while the Fuzzy controller has an execution time in the order of hundreds of micro-seconds. Even though, all the controllers manage to complete the respective control laws within the sampling period ($T_s = 0.01$s), it is preferable to use the PID or the LQR controllers since it can allow additional functionalities to be implemented on the same computing platform, such as the motion planning, localization algorithms, and so on.

VI. Conclusion and Future Work

To improve the evaluation process of vehicles ability to detect and classify bicycles, a driverless instrumented bicycle is designed with components which could fit into the bicycle frame or be hidden away in a bicycle basket. To evaluate and compare different control algorithms for stabilising the system, two different models are developed. The instrumented bicycle senses the lean angle and adjusts the steering angle to maintain balance. For the purpose of controlling the handlebar, two different tuned PID controllers, an LQR and a Fuzzy controller are evaluated and implemented in both simulations and on the instrumented bicycle. In the simulation, the controllers are compared both on a linear and a nonlinear model and a disturbance is induced in the lean angle measurements to evaluate the robustness of the four controllers. The outcome from the conducted experiments shows promising results when using the PID controllers or LQR, which all manages to keep the bicycle balanced for over 40 seconds on a narrow roller.

To maintain the silhouette of a bicycle, the next iteration of the instrumented bicycle should focus on embedding the components into the bicycle frame. Since the LQR and PID controller with the set of higher gains shows consistent results in simulation and experiments these controllers should be considered for balancing the bicycle in future work where path following will be addressed.

References

[1] P. Fairley, "Self-driving cars have a bicycle problem [news]," *IEEE Spectrum*, vol. 54, no. 3, pp. 12–13, 2017.

[2] N. H. Getz and J. E. Marsden, "Control for an autonomous bicycle," in *IEEE Int. Conf. on Rob. and Autom.*, 1995, pp. 1397–1402 vol.2.

[3] K. J. Åström, *et al.*, "Bicycle dynamics and control: adapted bicycles for education and research," *IEEE Control Systems Magazine*, vol. 25, no. 4, pp. 26–47, 2005.

[4] A. L. Schwab and J. P. Meijaard, "A review on bicycle dynamics and rider control," *Vehicle Syst. Dyn.*, vol. 51, no. 7, pp. 1059–1090, 2013.

[5] J. Tan, *et al.*, "Learning bicycle stunts," *ACM Trans. Graph.*, vol. 33, no. 4, 2014.

[6] M. Baquero-Suárez, *et al.*, "A robust two-stage active disturbance rejection control for the stabilization of a riderless bicycle," *Multibody System Dynamics*, no. 45, pp. 1–29, 2018.

[7] A. Shafiekhani, *et al.*, "Design and implementation of an adaptive critic-based neuro-fuzzy controller on an unmanned bicycle," *Mechatronics*, vol. 28, pp. 115 – 123, 2015.

[8] F. Whipple, *Stability of the Motion of a Bicycle*. Quarterly Journal of Pure and Applied Mathematics 30, 1899.

[9] J. P. Meijaard, *et al.*, "Linearized dynamics equations for the balance and steer of a bicycle: a benchmark and review," *Royal society A: mathematical, physical and engineering sciences*, vol. 463, no. 2084, pp. 1955–1982, 2007.

[10] H. D. Sharma and N. UmaShankar, "A Fuzzy Controller Design for an Autonomous Bicycle System," *IEEE Int. Conf. on Eng. of Int. Syst.*, pp. 1–6, 2006.

[11] J. Hauser, *et al.*, "Aggressive motorcycle trajectories," *IFAC Proceedings Volumes*, vol. 37, no. 13, pp. 949–954, 2004.

[12] D. Meehan and A. Ruina, "Linear and nonlinear controllers of an autonomous bicycle have almost identical basins of attraction in a non-linear simulation," in *Symp. on the Dyn. and Contr. of Single Track Vehicles*, 2020.

[13] M. R. García, *et al.*, "Stabilizing an urban semi-autonomous bicycle," *IEEE Access*, vol. 6, pp. 5236–5246, 2018.

[14] A. L. Schwab, *et al.*, "Some recent developments in bicycle dynamics and control," in *European Conf. on Structural Control (ECSC)*, 2008.

[15] J. He, *et al.*, "Constant-velocity steering control design for unmanned bicycles," in *IEEE Int. Conf. on Rob. and Biom.*, 2015, pp. 428–433.

[16] Y. Tanaka and T. Murakami, "Self sustaining bicycle robot with steering controller," in *IEEE Int. W. on Advanced Motion Control*, 2004, pp. 193–197.

[17] S. Vatanashevanopakorn and M. Parnichkun, "Steering control based balancing of a bicycle robot," in *IEEE Int. Conf. on Rob. and Biom.*, 2011, pp. 2169–2174.

[18] M. A. Anjumol and V. R. Jisha, "Optimal stabilization and straight line tracking of an electric bicycle," in *Int. Conf. on Power Signals Contr. and Comp.*, 2014, pp. 1–6.

[19] Y. Tanaka and T. Murakami, "A study on straight-line tracking and posture control in electric bicycle," *IEEE Trans. Int. El.*, vol. 56, no. 1, pp. 159–168, 2009.

[20] T. Andersson, *et al.*, "A loop shaping method for stabilising a riderless bicycle," in *European Conf. on Mobile Robots*, 2019, pp. 1–6.

[21] D. Bruijnen, *et al.*, "Optimization aided loop shaping for motion systems," in *IEEE Int. Conf. on Contr. Appl.*, 2006, pp. 255–260.

[22] E. V. Kumar, *et al.*, "Algebraic approach for selecting the weighting matrices of linear quadratic regulator," in *Int. Conf. on Green Comp. Comm. and El. Eng.*, 2014, pp. 1–6.

[23] G. F. Franklin, *et al.*, *Feedback Control of Dynamic Systems*, 7th ed. Pearson, 2014.

[24] Y. R. Abdolmalaki, "Implementation of Fuzzy Inference Engine for equilibrium and roll-angle tracking of riderless bicycle," *arXiv e-prints*, pp. 1–5, 2017.