Evolving Spiking Neural Networks to Mimic PID Control for Autonomous Blimps

Tim Burgers
Delft University of Technology
Delft, The Netherlands

Stein Stroobants
Delft University of Technology
Delft, The Netherlands
s.stroobants@tudelft.nl

Guido C.H.E. de Croon
Delft University of Technology
Delft, The Netherlands
g.c.h.e.decroon@tudelft.nl

***Abstract*—In recent years, Artificial Neural Networks (ANN) ave become a standard in robotic control. However, a signif- ant drawback of large-scale ANNs is their increased power onsumption. This becomes a critical concern when designing utonomous aerial vehicles, given the stringent constraints on ower and weight. Especially in the case of blimps, known or their extended endurance, power-efficient control methods re essential. Spiking neural networks (SNN) can provide a olution, facilitating energy-efficient and asynchronous event- riven processing. In this paper, we have evolved SNNs for ccurate altitude control of a non-neutrally buoyant indoor limp, relying solely on onboard sensing and processing power. he blimp's altitude tracking performance significantly improved ompared to prior research, showing reduced oscillations and a ninimal steady-state error. The parameters of the SNNs were ptimized via an evolutionary algorithm, using a Proportional- erivative-Integral (PID) controller as the target signal. We eveloped two complementary SNN controllers while examining arious hidden layer structures. The first controller responds wiftly to control errors, mitigating overshooting and oscillations, hile the second minimizes steady-state errors due to non- eutral buoyancy-induced drift. Despite the blimp's drivetrain mitations, our SNN controllers ensured stable altitude control, mploying only 160 spiking neurons.**

***Index Terms*—Spiking Neural Networks (SNN), Leaky- ntegrate-and-Fire model, Neuromorphic Control**

I. INTRODUCTION

Throughout history, humans have been fascinated by how nimals gracefully and precisely navigate complex environ- ents. This fascination has inspired efforts to understand e brain's computational processes behind such behavior, ading to the development of Artificial Neural Networks ANN). These ANNs represent the neural processes using implified mathematical models. Their ability to approximate omplex non-linear functions makes them highly effective controlling complex systems, such as quadrotors [1; 2]. Iowever, as the size of ANNs grow, both response latency and omputational demands increase. The latter poses particular hallenges for robotic applications with restricted onboard nergy capacity, such as flying robots. A solution may be ound in the information transmission methods employed by iological brains. ANNs rely on continuous-valued signals, hereas the biological brain employs sparse spatial-temporal "spike" signals—brief, rapid increases in neuron voltage—for data encoding and transmission.

This material is based upon work supported by the Air Force Office of cientific Research under award number FA8655-20-1-7044.

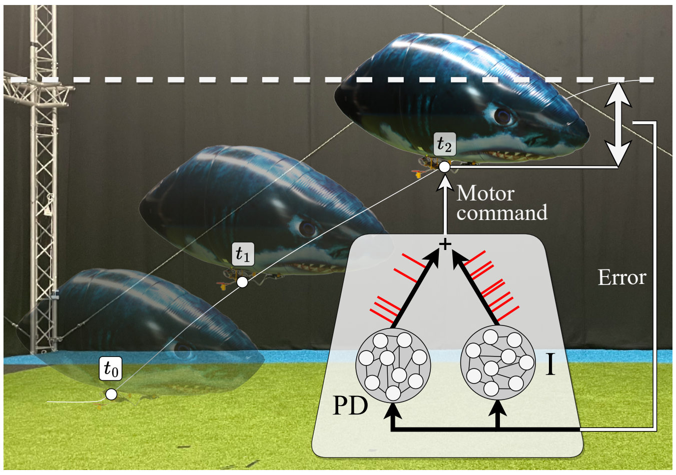


Fig. 1. The proposed SNN altitude controller for the blimp, where $[t_0,t_1,t_2]$ indicates time instances of the blimp's altitude while approaching a setpoint, marked by the dashed line.

There are neural networks that use this spike-based approach to transmitting information, called a Spiking Neural Network (SNN) [3]. These more biologically plausible neural networks show potential for energy-efficient and low-latency controllers [4]. This fact was demonstrated in a study by Vitale et al. [5], where an SNN controller in a fully neuromorphic control loop outperformed a conventional control loop on power consumption and control latency when tracking the roll angle of a bench-fixed 1 DoF birotor. However, the application of SNN controllers in robotics is still in its early stages of development. One of the biggest challenges is the availability of suitable training algorithms. The SNN's temporal dynamics, sparse spiking activity, and non-differentiable spike signal make most existing ANN training algorithms unsuitable [6]. Recent research has enabled the use of error backpropagation for SNNs by means of surrogate-gradient algorithms [7]. Nevertheless, training SNNs using these gradient-based algorithms is still difficult due to the susceptibility to local minima, exploding/vanishing gradient, and sensitivity to initial conditions [8].

Due to the increased training complexity of SNNs compared

› non-spiking counterparts, practical applications of SNNs ı the robotic control domain are still limited. Designing fully neuromorphic SNN controller to emulate low-level ontrollers, such as the Proportional-Integral-Derivative (PID) ontrol, still remains a complex task. Recent research showed NNs performing differentiation and integration within the etwork, by manually configuring neuron connections and ⁄eights [9; 10; 11]. In these studies, the controllers were nplemented on Intel's Loihi neuromorphic processor, show- asing the promise of neuromorphic hardware by exhibiting ery low latencies [12]. In Zaidel et al. [13], multiple popu- ıtions with pre-determined network parameters were used to nplement all three pathways of the PID controller to control 6 Degrees of Freedom (DoF) robotic arm. The integral athway was implemented using a fully recurrent population f neurons, while differentiation was achieved by using a low and fast time constant for two populations. Although ıe integral controller succeeded in reducing the steady-state rror, it was unable to completely eliminate it. In another study, piking neurons were trained to replace the rate controller of tiny quadrotor, the Crazyflie [14]. Integration in the SNN ontroller was achieved through discretized Input-Weighted 'hreshold Adaptation (IWTA), where the threshold depends n the previous layer's spiking activity. The training process ad limitations because it relied partially on predetermined onnections and grouped network parameters.

In this work, we investigate the different mechanisms, recur- ɛncy and IWTA, used in prior SNN PID research that enabled ifferentiation and/or integration. For each mechanism, we volve an SNN to control the altitude of a real-world indoor limp. The blimp is an interesting test platform for the SNN ontroller, allowing validation of all components of the PID ontroller. The changing buoyancy over time requires a good ıtegrator to be present. Moreover, due to the high delays and low system dynamics of a blimp, a high proportional gain ; necessary in the reference PID controller. This reduces the limp's rise time, requiring a strong derivative in the con- ·oller's output to prevent overshoot and oscillations. In Gon- alez et al. [15], an open-source indoor blimp was designed nd used as a test vehicle for an evolved neuromorphic altitude ontroller, showing adequate tracking of the reference signal. Iowever, even after including an additional non-spiking PD ontroller to the output of the SNN controller, there were still scillations present of approximately ± 0.3m. Additionally, ıe SNN was only trained on a neutrally buoyant blimp. Slight hanges in the buoyancy of the blimp would cause a steady tate error, which the controller was unable to eliminate.

We build further on this research, presenting here the ›llowing contributions: 1) We developed a fully neuromorphic eight controller for a blimp (visualized in Figure 1), using an volved SNN of only 160 neurons that is able to minimize the vershoot and oscillations while also removing the steady-state rror caused by the buoyancy of the blimp. 2) We analyze the ıdividual and combined influence of recurrent connections nd IWTA on the performance of the SNN controller 3) We ıade improvements to the hardware components of the open-source blimp, improving the onboard computational power and increasing the accuracy of the height measurements.

II. Methodology

The SNN controller consists of three layers of neurons, where all parameters are optimized using an evolutionary algorithm to mirror the output of a tuned PID controller. The Proportional-Derivative (PD) controller's rapid dynamics demand fast time constants, while the integral controller relies on slower dynamics and, thus, slow time constants. To facilitate the learning process, we split the controller into two separate parts based on the required time constants to model each component. After completing the training process, the evolved controllers are used to control the altitude of a helium-filled blimp. Detailed discussions on the SNN's structure, parameters, experiment setup, and the evolutionary training algorithm used in this study follow below.

A. Spiking Neural Network Controller

We use current-based Leaky-Integrate and Fire (LIF) neurons with a soft reset for the threshold (ϑ). The discretized equations that describe the dynamics of the three states of the neuron (synaptic current $i(t)$, membrane potential $v(t)$ and spike train $s(t)$) are described as follows:

$$i_i(t) = \tau_i^{\text{syn}} i_i(t-1) + \sum W_{ij} s_j(t) \tag{1}$$
$$v_i(t) = \tau_i^{\text{mem}} v_i(t-1) + i_i(t) - s_i(t-1)\vartheta_i \tag{2}$$
$$s_i(t) = H\left(v_i(t) - \vartheta_i\right) \tag{3}$$

where subscript i and j denote the post- and presynaptic neuron respectively. The discretized time constants, known as the decay parameters, of the synapses and the membrane potential are respectively referred to as τ^{syn} and τ^{mem}. The spiking behavior of a neuron is modeled using the Heaviside step function H, which outputs a spike when the membrane potential exceeds the threshold.

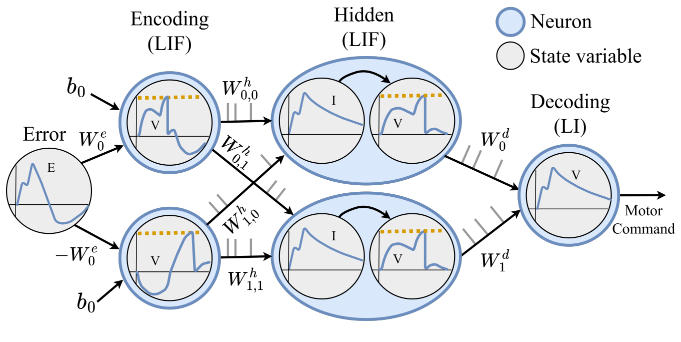
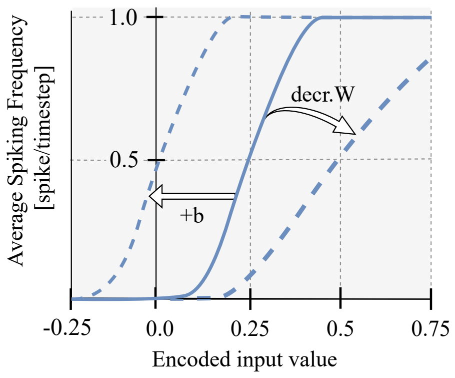


Fig. 2. The basic structure of the SNN controller. The encoding layer has an additional bias (b) added to the input current.

The basic structure of a spiking neural network controller is schematically depicted in Figure 2. The input weights, indicated by W^e, W^h and W^d, are linked to the encoding, hidden and decoding layers, respectively. The encoding layer is responsible for translating the floating-point input into sequence of spikes, and conversely, the decoding layer erforms the reverse operation.

The input to the network is the error of the controlled tate. After applying the encoding weights and biases, the rror signal is directly used as the synaptic current for the ncoding neuron ($\tau^{syn} = 0$). To facilitate the training of the ncoding layer, we paired neurons with shared bias and flipped /eight sign. This results in symmetric encoding, ensuring imilar spiking patterns for both positive and negative errors. 'he effect that the input weight and bias have on the spiking ehavior of a LIF neuron is presented in Figure 3. The ncoding layer incorporates a bias to achieve spike activity or the encoding of error values close to zero, which would therwise be impossible.



ig. 3. Influence of the input weight and bias on the spiking frequency of a IF neuron.

We study the effect of using different types of structures for 1e hidden layer of the SNN controller in this work. The most asic type of hidden layer (LIF) is depicted in Figure 2. In ddition to the basic structure, we evaluated the influence of ecurrency [13; 16] and Input-Weighted Threshold Adaptation [WTA) [14], as both these network structures demonstrated 1eir essential role in enabling integration within the SNN. We ocused solely on threshold adaptation linked to the incoming piking activity rather than the hidden neuron's activity itself. An overview of the different hidden layer structures is shown 1 Figure 4. In contrast to the original implementation in [14], 1e threshold for the IWTA-LIF neurons is modeled using a ecay term (τ^{th}), where the threshold converges back to a ase value (ϑ), after an increase/decrease (W^{th}) caused by an 1coming spike. This method of implementing IWTA adds new ynamics to the threshold and increases the solution space.

The spiking neural network controller is decoded using a ingle leaky-integrator neuron, that calculates the exponential 1oving average of the spikes in the hidden layer.

'. *Real-World Experiment*

To validate the performance of the SNN controller on real-world application, we implemented an SNN altitude ontroller for an open-source micro-blimp developed in [15]. 'he combination of the buoyancy-caused drift and slow ystem dynamics make the blimp a useful test vehicle for

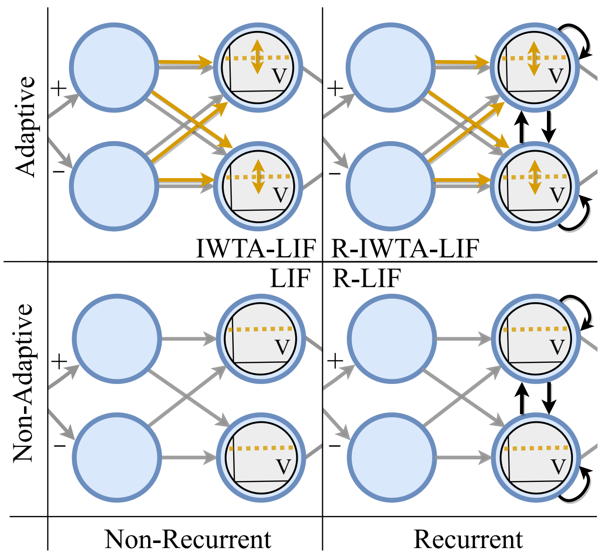


Fig. 4. Visualization of the different hidden layer structures. The two left and right circles represent the encoding and the hidden neurons respectively.

this research. The input of the SNN height controller is the difference between a reference altitude and the onboard lidar measurement, which makes it a fully on-board closed-loop control system. The output of the SNN is the motor command, $u \in [-3.3, 3.3]$, where $u < 0$ indicates downward and $u > 0$ upwards movement. The blimp's control system consists of two coreless direct current (DC) motors attached to a 180-degree rotating shaft, which enable the rotors to push the blimp both up- and downwards. A visual representation of the blimp and its hardware components is provided in Figure 5. Two improvements have been made to the blimp's hardware setup. Firstly, to improve the altitude tracking ability, we implemented a LiDAR sensor, the TFmini S LiDAR-module, which significantly increased the accuracy from ± 20cm to ± 1cm. Secondly, the Raspberry Pi Zero 2 W, running on Ubuntu 20.04, replaced its predecessor to increase the processing power and prevent software compatibility issues. The total weight of all hardware components attached to the blimp is 140g. In order to ensure smooth communication between all system components, we have used the Robot Operating System (ROS1) framework. The control loop runs at a rate of 10 Hz.

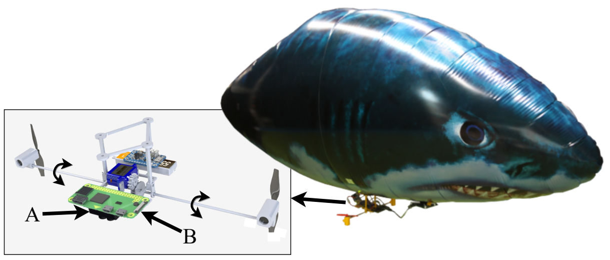


Fig. 5. The open-source micro-blimp used for the real-world experiments [15]. Two adjustments made on the blimp are: A) TFmini S LiDAR-module B) Raspberry Pi Zero 2 W.

:. *Evolutionary Training Algorithm*

The training process for the SNN controller employs an volutionary algorithm, consisting of two recurring steps: poplation creation and evaluation, with each cycle representing ne generation of the evolution. In the first step, a set number f individuals forms the population, each representing a unique ontroller with slightly varying parameters. The second step anks these individuals based on performance via a cost unction, and this information guides the creation of the new opulation. Further details for each step are discussed below.

Every training loop was executed on the DelftBlue superomputer, running on 20 cores for 50,000 generations with 50 ıdividuals [17]. To prevent overfitting, we randomly sampled ne of the 100 training datasets to evaluate each generation. ıdditionally, both the size and the frequency of the step inputs sed in each dataset were varied to enhance the diversity of ıe training data. We conducted a total of 30 training loops ɔr each combination of controller type (2) and hidden layer tructure (4), resulting in a total of 240 training sessions. Each ontroller is limited to 80 neurons to showcase the potential f small-scale networks for neuromorphic control.

1) Population Creation: For the creation of a new popuıtion, we have used a Covariance Matrix Adaptation Evoluonary Strategy (CMA-ES) [18]. CMA-ES is a distributionased optimization algorithm that iteratively adjusts the mean nd covariance matrix of a multi-variate Gaussian distribution, :om which all network parameters of each individual are ampled.

The CMA-ES is implemented using the *Evotorch* Python brary [19]. Every network parameter that needs to be evolved ; initialized using the mean and standard deviation of a ǐaussian distribution. Strict-bounded parameters, such as the ecay constants in the neuron model, are constrained by ejecting and resampling. The initial mean is set by sampling :om a uniform distribution within parameter bounds, while the tandard deviation is set to 1/10th of the parameter's range. ın overview of all trained parameters including the bounds ; provided in Table I, where W^r represents the recurrent /eights.

TABLE I
OVERVIEW OF ALL PARAMETERS AND BOUNDS USED TO EVOLVE THE SNN, WITH ADDITIONAL PARAMETERS IN THE HIDDEN LAYER FOR *RECURRENCY AND †IWTA

	Param.	Size	Bounds
Enc.	W^e	N	[-2, 2]
	b	N	[-1, 1]
	τ^{mem}	N	[0,1]
	ϑ	N	[0,10]
Dec.	W^d	N	[-1,1]
	τ^{mem}	N	[0,1]

	Param.	Size	Bound
Hidden	W^h	2N	[-2, 2]
	τ^{syn}	N	[0, 1]
	τ^{mem}	N	[0, 1]
	ϑ	N	[0, 10]
	$^*W^r$	NxN	[-1, 1]
	$^\dagger\tau^{th}$	N	[0,1]
	$^\dagger W^{th}$	NxN	[-1, 1]

2) Population Evaluation: The performance of each indiidual SNN controller is determined by comparing the output f the SNN controller (u) to the output of a tuned PD r integral controller ($\hat{u}$). The SNN is tasked to learn the ıapping between the input signal (e) to the output of the target controller ($\hat{u}$). The discretized equation that describes the PID response is provided below:

$$\hat{u_k} = \underbrace{K_p e_k + K_d \frac{e_k - e_{k-1}}{T}}_{\text{PD controller}} + \underbrace{K_i (\sum_{k=0}^{k} T e_k)}_{\text{Integral controller}} \tag{4}$$

where K_p, K_i, and K_d refer to the proportional, integral and derivative gains respectively and T is denoted by the sampling period. Both the input error signal and the target controller response are recorded in a dataset.

To quantify the performance of the SNN compared to the target controller, the Mean Absolute Error (MAE) was used as the main term in the cost function. The MAE was used instead of the mean square error (MSE) to prevent over-penalization of the error that is created during the transient response to a step input because this led to more oscillations in the steady state. Additionally, the cost function was augmented with the Pearson Correlation Coefficient (PCC) [20], denoted by ρ, to incentivize that the sign of the output of the SNN and PD/I controllers is equal. The PCC measures the linear correlation between two signals, ranging from -1 to 1, where the latter indicates a fully linear relation. Since the fitness function is a minimization function, the PCC is included by adding $1 - \rho$ to the MAE. This results in the following cost function that was used in the population evaluation step of the evolutionary training process:

$$L(u, \hat{u}) = \text{MAE}(u, \hat{u}) + (1 - \rho(u, \hat{u})) \tag{5}$$

D. *Dataset Generation*

The methods used to gather the test and training data for each controller are discussed below.

1) PD controller: To successfully train the PD SNN controller, we need a broad spectrum of error signals. Therefore, we used a semi-randomly tuned PID controller on the neutrally buoyant real-world blimp. The error signal of these recordings is used as the SNN input data of the dataset. The target signal for the training algorithm is generated by passing the recorded error signal through a PD controller with the tuned gains for the blimp's altitude controller.

2) Integral controller: The Integral SNN has to learn to integrate the error within the SNN itself. For this training process, the decay parameter of the decoding neuron is purposely bound to ensure a quick decay (eg. [0-0.3]) in order to prevent the algorithm from converging to a slow decay. A slow decoding decay parameter would imply that the integration only happens in the decoding neuron, instead of in the hidden layer.

If the buoyancy remains constant throughout the training process of the integral controller, the network might learn to add a bias to counteract the buoyancy. Therefore we must adjust the buoyancy during training to ensure that the network learns to integrate information over time. Instead of recording multiple datasets with varying buoyancy, we decided to train the Integral controller on a model where we could also change ıe ”buoyancy” within a single dataset to facilitate the learning rocess.

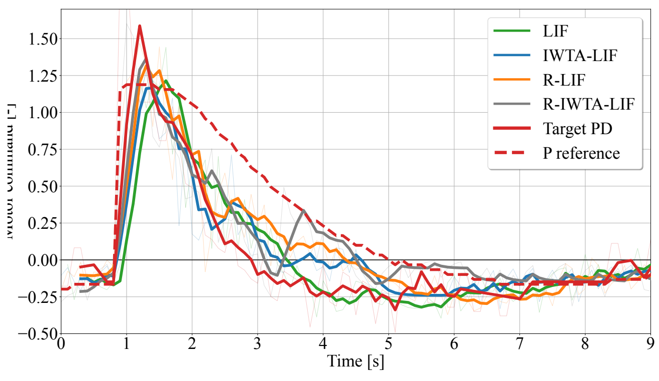
The blimp was modeled by the double integrator control roblem and controlled using a PID controller. The integral ain was set to match that of the tuned real-world blimp, while ıe PD gains were adjusted to align the model’s dynamics pproximately with those of the tuned PID-Blimp system. In ıe double integrator system, the output of the controller, $u(t)$; directly proportional to the second derivative of the state lus an additional bias: $\ddot{x}(t) = u(t) + b$. The bias simulates ıe level of buoyancy of the blimp. Every time a step input ; received, the bias is randomly sampled ($U(-4, 4)$). Each ataset consists of 5 step inputs maintained for 50s.

III. Results

The first subsection displays the training outcome of both NN controllers using a test dataset, followed by the assess- ıent of their performance on an actual blimp. The results also ontain a performance analysis of various neural mechanisms ⁄ithin the hidden layer.

. *Training of the SNN Controllers*

1) PD SNN Controller: The result of the PD training rocess is provided in Figure 6, showing a single step response :om the test dataset. The solid red line represents the SNN’s ırget, while the dashed red line depicts the proportional con- ·oller. The P controller is included to visualize the additional ffect of the derivative. Initially, all spiking PD controllers how clear influences of the derivative, as they match the target ignal. However, after the initial damping of the proportional utput, the controllers start to diverge. To prevent overshoot nd oscillations, the derivative controller should counteract ıe proportional controller when the state is approaching the etpoint, which happens around 4 seconds in the Figure. The nly controller that counteracts the P controller sufficiently ; the LIF SNN. Based on this analysis and the lowest loss alue across the entire test dataset, shown in Table II, we pted to use the LIF SNN for the blimp’s altitude controller. ’he larger solution space for the recurrent and IWTA neuron tructures makes the search space more complex and leads to ɔcal minima.



ig. 6. The moving average of the step responses of all evolved SNN ɔntrollers compared to a PD target signal using a test dataset. The LIF shows ɛrivative action by damping the command.

TABLE II
Loss for SNN PD controllers (u) on complete test dataset (500s) using tuned PD as target ($\hat{u}$)

	LIF	IWTA-LIF	R-LIF	R-IWTA-LIF
$L(u, \hat{u})$	0.68	0.74	0.74	0.73

2) Integral SNN Controller: The result of the integral training process is provided in Figure 7, showing the response of the different hidden layer mechanisms to a test dataset. The LIF SNN failed to learn to integrate, hence, it is excluded from the figure. All three spiking controllers following the test signal, without a clear standout performer. To see how well the controllers perform in the real-world, they are all tested on the indoor blimp.

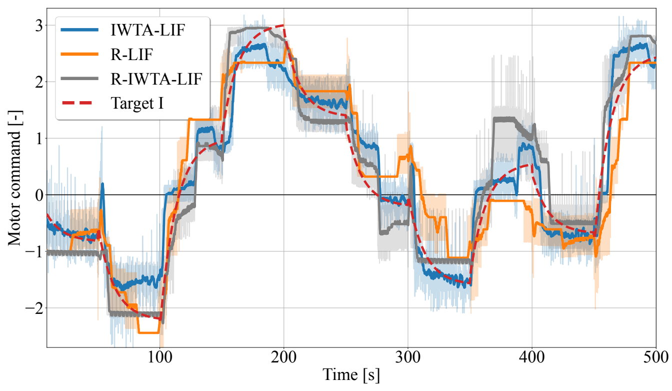


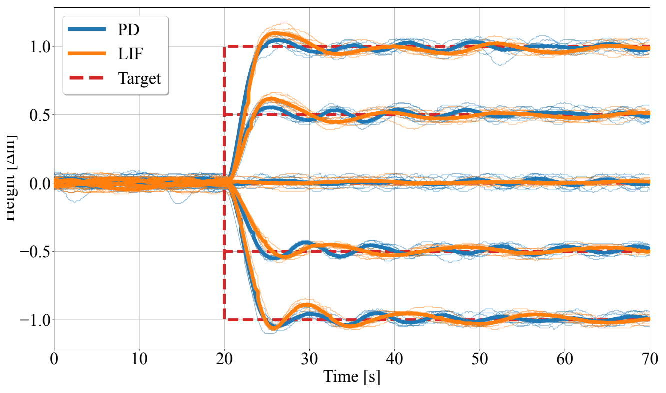
Fig. 7. The moving average of the step responses of three evolved SNN controllers compared to a integral target signal using a test dataset, with a changing bias every step input.

B. *Performance of SNN controlling Real-World Blimp*

To analyze the performance of each controller separately, we first test the PD SNN controller using a neutrally buoyant blimp. Afterward, we added some weight to the blimp to make it negatively buoyant. The negatively buoyant blimp is used first to evaluate the performance of the SNN I controllers, followed by the evaluation of the fully spiking controller.

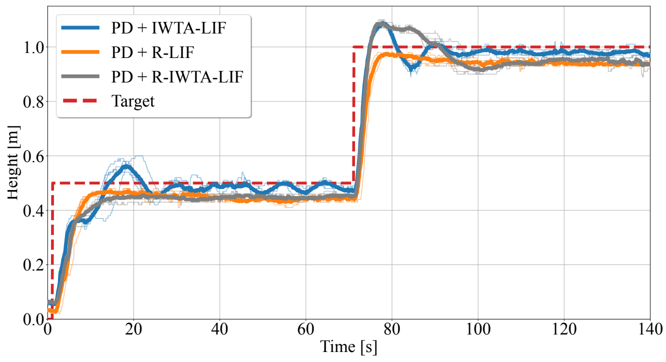
1) PD control of neutrally buoyant blimp: We assess the performance of the PD SNN controller with LIF hidden structure by comparing it to a tuned conventional PD controller. The tracking accuracy is tested using five different step sizes Δh=[1,0.5,0,-0.5,-1], maintained for 50s each and the results are shown in Figure 8. Both the conventional PD and the SNN show small oscillations, $\pm$6 cm, around the setpoint. These oscillations are caused by the discretized mapping of the motor command to the actual voltage sent to the motors using PWM. This causes a deadzone to be present in the motor command signal, which is the region of the motor commands, u =[-0.1,0.1], that does not result in the actuation of the rotors.

2) Integral control of negatively buoyant blimp: To isolate the effect of the spiking integrator, we used the non-spiking PD controller in combination with the spiking integrator to control the negatively buoyant blimp. The result of two-step responses



ig. 8. Real-world step responses of the altitude of the neutrally buoyant limp using a conventional PD control and an SNN controller with no ecurrency or IWTA in the hidden layer. Each setpoint was tested eight times or every controller and the average is shown by the thick line.

s shown in Figure 9, where the setpoint was changed after 70 econds. When analyzing the steady-state error, we take the verage over the last 10 seconds of each step. The steady-state rror of the IWTA-LIF (±2 cm) is significantly smaller than ne ones for the R-LIF and R-IWTA-LIF (±5 cm). Despite ne small oscillations caused by the LIF SNN, we decided to se this spiking integrator for the full spiking controller due o the minimal steady-state error.



ig. 9. Real-world step responses of the altitude of the negatively buoyant limp using a conventional PD controller and different spiking integrators. he average over five runs is shown for each controller.

3) Full spiking control of negatively buoyant blimp: The ombination of both the spiking PD (LIF) and integral (IWTA-IF) controller is shown in Figure 10. We analyzed the step reponse for the blimp using different setpoints h=[0.5,1,1.5]m, naintained for 70s. The combined SNN controller demontrates effective altitude control while minimizing the steadytate error to ±3cm. However, the SNN controller shows elatively large initial oscillations when receiving a downward tep input, compared to the upward steps. The oscillations esult from the delay introduced when the rotors must make a 80-degree turn during direction changes. Given the blimp's egative buoyancy, continuous upward thrust is essential for tability. In cases of upward step inputs, the rotors constantly push the blimp upwards. Conversely, when a lower setpoint is used, the blimp is initially pushed downwards by the rotor before pushing back up to attenuate the movement. This difference in overshoot is also visible for the PID controller, with an average overshoot of 14cm for upward steps and 20cm for downward steps.

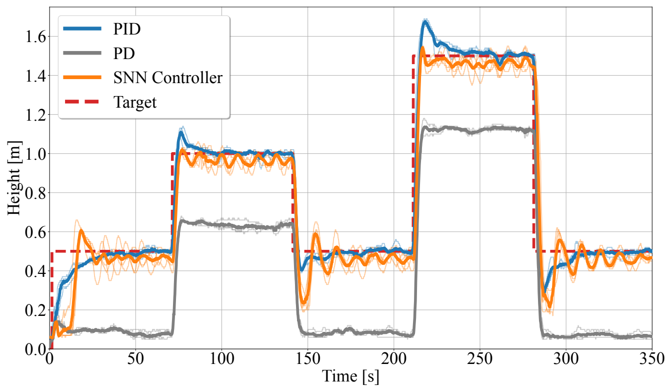


Fig. 10. Real-world multi-step response of the altitude of the negatively buoyant blimp using a non-spiking PID & PD controller and the SNN controller, which is the combination of the SNN PD (LIF) and the SNN integral (IWTA-LIF) controller. The average over five runs is shown for each controller.

IV. CONCLUSION

In this work, we evolved a spiking neural network (SNN) that successfully controls the altitude of a non-neutrally buoyant indoor blimp. The SNN parameters were optimized through an evolutionary algorithm, facilitating extensive exploration of the solution space. This exploratory training approach allowed for an in-depth analysis of various hidden-layer configurations, recurrency and Input Weighted Threshold Adaptation (IWTA), for the Leaky-Integrate and Fire (LIF) neuron model. As a result, we developed two complementary SNN controllers which, when combined, achieved accurate tracking of the reference state. The first controller exhibited rapid response to control errors while effectively mitigating overshoot and large oscillations, after being trained on a tuned PD controller for the blimp. In parallel, the second controller was designed to minimize steady-state errors arising from the blimp's non-neutral buoyancy-induced drift. This controller learned to perform integration of the error using IWTA within the hidden layer of the network.

Despite the limitation within the blimp's current drivetrain configuration, the developed SNN controllers have showcased their ability to maintain stable altitude control, employing just 160 spiking neurons. All processing and sensing is performed onboard the blimp, with the SNN running on the Raspberry Pi's CPU. Future research will focus on the completion of the neuromorphic control loop, integrating event-based sensors with neuromorphic processors. This integration aims to fully demonstrate the potential of neuromorphic computing in robotic control.

REFERENCES

[1] D. Zhang, A. Loquercio, X. Wu, A. Kumar, J. Malik, and M. W. Mueller, “A Zero-Shot Adaptive Quadcopter Controller,” 9 2022.

[2] M. Heryanto, H. Suprijono, B. Yudho, and B. Kusumoputro, “Attitude and altitude control of a quadcopter using neural network based direct inverse control scheme,” *Advanced Science Letters*, vol. 23, pp. 4060–4064, 05 2017.

[3] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[4] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, “A survey of robotics control based on learning-inspired spiking neural networks,” *Frontiers in Neurorobotics*, vol. 12, 2019.

[5] A. Vitale, A. Renner, C. Nauer, D. Scaramuzza, and Y. Sandamirskaya, “Event-driven Vision and Control for UAVs on a Neuromorphic Chip,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2021-May, pp. 103–109, Institute of Electrical and Electronics Engineers Inc., 2021.

[6] A. Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity, “A review of learning in biologically plausible spiking neural networks,” *Neural Networks*, vol. 122, pp. 253–272, 2020.

[7] E. O. Neftci, H. Mostafa, and F. Zenke, “Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-based optimization to spiking neural networks,” *IEEE Signal Processing Magazine*, vol. 36, pp. 51–63, 11 2019.

[8] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, “A survey of robotics control based on learning-inspired spiking neural networks,” *Frontiers in Neurorobotics*, vol. 12, 2019.

[9] R. K. Stagsted, A. Vitale, A. Renner, and L. B. Larsen, “Event-based PID controller fully realized in neuromorphic hardware: A one DoF study,” in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, pp. 10939–10944, 7 2020.

10] R. K. Stagsted, A. Vitale, A. Renner, and L. B. Larsen, “Towards neuromorphic control: A spiking neural network based PID controller for UAV,” in *Robotics: Science and Systems XVI*, vol. 1, Robotics: Science and Systems Foundation, 1 2020.

11] S. Stroobants, J. Dupeyroux, and G. De Croon, “Design and implementation of a parsimonious neuromorphic PID for onboard altitude control for MAVs using neuromorphic processors,” in *ICONS’22 Proceedings*, vol. 2657, pp. 1–9, CEUR-WS, 2022.

12] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. H. Weng, A. Wild, Y. Yang, and H. Wang, “Loihi: A Neuromorphic Manycore Processor with On-Chip Learning,” *IEEE Micro*, vol. 38, pp. 82–99, 1 2018.

[13] Y. Zaidel, A. Shalumov, A. Volinski, L. Supic, and E. Ezra Tsur, “Neuromorphic NEF-Based Inverse Kinematics and PID Control,” *Frontiers in Neurorobotics*, vol. 15, 2 2021.

[14] S. Stroobants, C. De Wagter, and G. C. H. E. de Croon, “Neuromorphic Control using Input-Weighted Threshold Adaptation,” vol. 1, Association for Computing Machinery, 2023.

[15] M. Gonzalez-Alvarez, J. Dupeyroux, F. Corradi, and G. C. De Croon, “Evolved neuromorphic radar-based altitude controller for an autonomous open-source blimp,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 85–90, 2022.

[16] H. Qiu, M. Garratt, D. Howard, and S. Anavatti, “Towards crossing the reality gap with evolved plastic neurocontrollers,” in *GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, vol. 1, pp. 130–138, Association for Computing Machinery, 6 2020.

[17] Delft High Performance Computing Centre (DHPC), *DelftBlue Supercomputer (Phase 1)*, 2022. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1.

[18] N. Hansen, “The CMA Evolution Strategy: A Tutorial,” pp. 1–39, 2016.

[19] N. E. Toklu, T. Atkinson, V. Micka, P. Liskowski, and R. K. Srivastava, “EvoTorch: Scalable Evolutionary Computation in Python,” pp. 1–25, 2023.

[20] J. Benesty, J. Chen, Y. Huang, and I. Cohen, “Pearson Correlation Coefficient,” in *Noise Reduction in Speech Processing*, pp. 1–4, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.