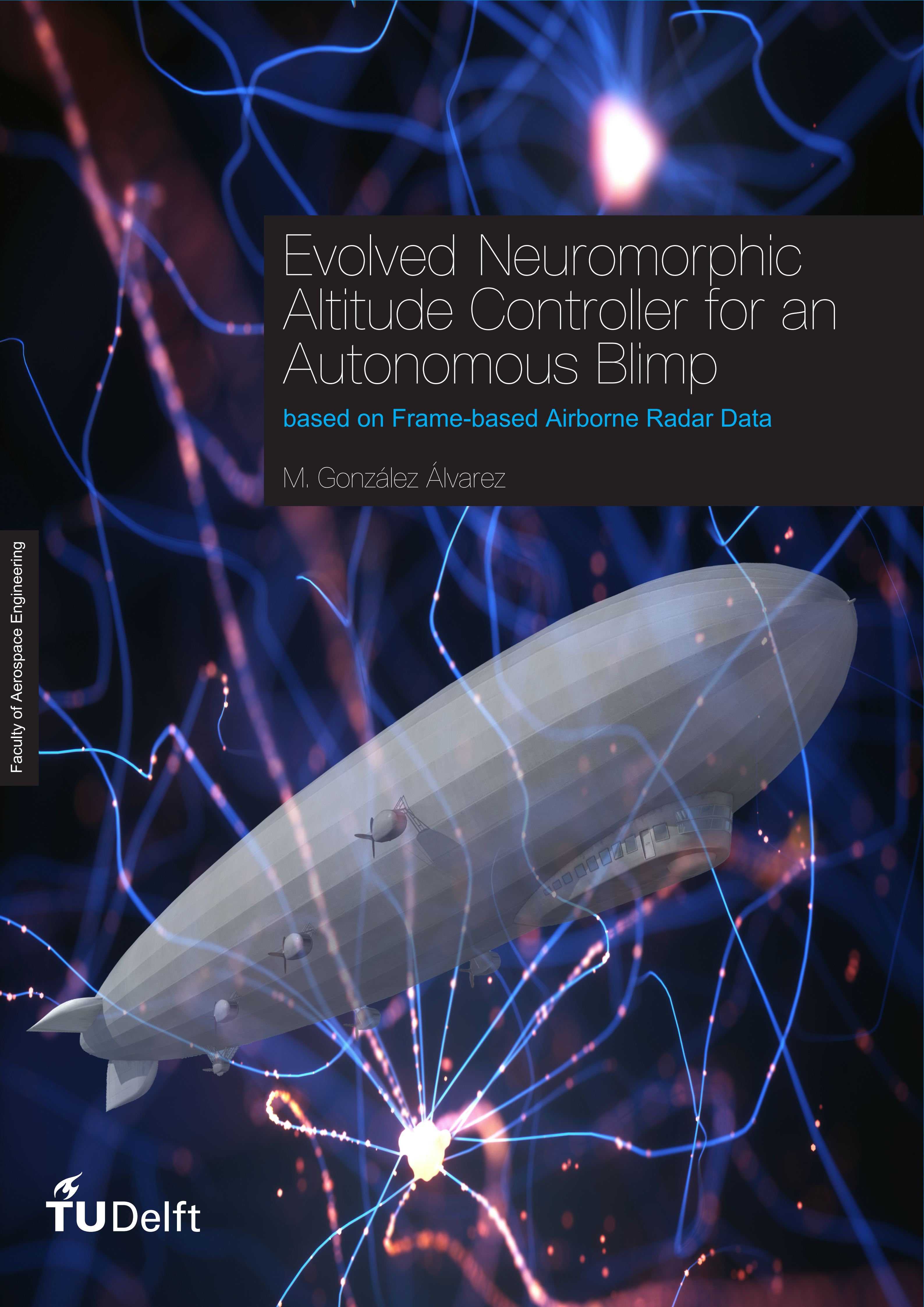


Evolved Neuromorphic Altitude Controller for an Autonomous Blimp

based on Frame-based Airborne Radar Data

M. González Álvarez

Faculty of Aerospace Engineering

TU Delft

Evolved Neuromorphic Altitude Controller for an Autonomous Blimp

based on Frame-based Airborne Radar Data

by

M. González Álvarez

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday November 2, 2021 at 9:00 AM.

Student number:	5044030	
Thesis committee:	Prof. dr. J. Dupeyroux,	TU Delft, supervisor
	Prof. dr. G.C.H.E. de Croon,	TU Delft, supervisor
	Dr. F. Corradi,	imec, supervisor
	Prof. dr. F. Fioranelli,	TU Delft, examiner

TU Delft

Acknowledgements

First of all, I would like to thank Julien, Federico and Guido for your valuable guidance throughout this thesis' journey. It has been such an enriching learning experience in many different ways.

A word of appreciation also goes to Freek, for your great friendship, for all the fun, the bouldering, the beers and the walks around Delft.

Thanks also to Jenny and Maurizio from the Italian place, for all the tasty pasta, lasagna, risotto and chicken that kept me going.

Thanks to Delft, for becoming my home during the last two years, and for your fairy-tale-like beauty.

And last, but not least, thanks to the financial support from ”la Caixa” Foundation, who has selected me as a grantee of their Postgraduate Fellowships in Europe. It has been such a great honour to be part of your program.

Finally, with the reader's permission, the following words of thanks are in Spanish:

A Amaya, mi profesora de violín y una segunda madre para mí. Desde junio eres mi estrella y te llevaré siempre conmigo dondequiera que vaya.

A Tito Gonzalo, Tita Ángeles y Tía Eli.

A Mamá, Papá, Tita Candela y Emi, por todo vuestro cariño y amor incondicional. Gracias por las videollamadas de cada domingo en plena pandemia.

A Miguel, por tu increíble apoyo, tus infinitas horas de compañía real y virtual, tus visitas a Delft y Zúrich y tu cariño.

A Meri, por estar siempre ahí sin importar la distancia.

A Musi (Negrito), que siempre me saluda de la manera más adorable cuando voy a Caces. A Michaela y Maranella, las gatitas de Altstetten, por sacarme una sonrisa en momentos de estrés.

A Asturias, por tu verdor y tu lluvia, y Barcelona, por tu luz y tu sol. A las dos, por vuestros manjares deliciosos, la gente que contenéis, por ser casa y alegrarme el alma.

A la vida, porque sí.

M. González Álvarez
Barcelona, October 2021

Abstract

Micro robotic airships offer significant advantages in terms of safety, mobility, and extended flight times. However, their highly restrictive weight constraints pose a major challenge regarding the available computational power to perform the required control tasks. Thus, spiking neural networks (SNNs) are a promising research direction. By mimicking the biological process for transferring information between neurons using spikes or impulses, they allow for low power consumption and asynchronous event-driven processing. In this work, we propose an evolved altitude controller based on a SNN for an airship which relies solely on the sensory feedback provided by an airborne radar sensor. Starting from the design of a a lightweight, low-cost, open-source airship, we also present a low-control-effort SNN architecture, an evolutionary framework for training the network in a simulated environment, and a control scheme for ameliorating the performance of the system in real-world scenarios. The system's performance is evaluated through real-world experiments, demonstrating the advantages of our approach by comparing it with an artificial neural network (ANN) and a linear controller (PID). The results show an accurate tracking of the altitude command while ensuring efficient management of the control effort. The main contributions of this work are presented in the scientific paper, corresponding to Part I of the document. Besides the research on altitude control based on SNNs and their comparison with an ANN and a PID, this thesis includes an in-depth review of the relevant literate on the main topics covered, in Part II. Finally, a detailed explanation of the methodologies used, the conclusions and recommendations for future work are proposed in Part III.

Acronyms

ADC	Analog to Digital Converter
ANN	Artificial Neural Network
AoA	Angle of Arrival
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CW	Continuous Wave
DC	Direct Current
DL	Deep Learning
DNN	Deep Neural Network
DoF	Degree of Freedom
EMA	Exponential Moving Average
FFT	Fast Fourier Transform
FM	Frequency Modulation
FMCW	Frequency Modulated Continuous-Wave
GNSS	Global Navigation Satellite System
HH	Hodgin and Huxley
HHT	Hodgin and Huxley with Tables
IMU	Inertial Measurement Unit
IF	Intermediate Frequency
IR	Infrared
LIDAR	Light Detection And Ranging
LIF	Leaky Integrate and Fire
LMA	Linear Moving Average
LNA	Low-Noise Amplifier
LTA	Lighter-Than-Air
LSTM	Long-Short Term Memory
MAV	Micro Air Vehicle
MF	Median Filter
NN	Neural Network
PA	Power Amplifier
PD	Proportional Derivative
PI	Proportional Integral
PID	Proportional Integral Derivative
PRF	Pulse Repetition Frequency
PSP	Postsynaptic Potential
RC	Radio Controlled
ReLU	Rectifier Linear Unit
SRM	Spike Response Model
SSH	Secure Shell
STDP	Spike-Timing-Dependent-Plasticity
ReSuMe	Remote Supervised Method
RG	Reality Gap
RL	Reinforcement Learning

RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROS	Robot Operating System
RX	Receiver
SAR	Synthetic Aperture Radar
TX	Transmitter
UAV	Unmanned Aerial Vehicle

Contents

List of Figures

List of Tables

1 Introduction to the research

Even though the golden era of giant cargo airships has faded, the advantages offered by lighter-than-air crafts prevail. Their extended flight times, excellent ease of assembly, low acoustic footprint, low power consumption, and, consequently, reduced costs also pose complementary advantages to those that rotorcrafts have to offer [102]. This is why blimps are, slowly but surely, attracting increasing interest in the realm of unmanned aerial vehicles.

Robotic airships present endless possibilities in terms of their design, shape, and materials. For example, a three-propeller, low-cost platform equipped with a camera and a compact gondola built around the electronic components is presented in [66]. The authors also study the lift capabilities of different envelope materials over time, being Microfoil the one which provides the best results. An alternative design is proposed in [49], where a simpler gondola together with a novel actuation mechanism based solely on two propellers mounted on a rotating shaft, oriented using a servomotor, is introduced. Other examples of higher complexity include: a blimp made or four partial envelopes and four propellers [171], a configuration consisting on three light-weight servomotors and six propellers [121], or an omnidirectional spherical airship with four actuation units and a high-resolution camera [29].

Although these design alternatives have proven successful for their specific applications, they lack the versatility that can be achieved by leaving room for incorporating additional sensors and/or actuators. Besides, only [66] is open-source and lightweight enough to be mounted on commercially available Microfoil blimps. In this paper, we develop a versatile, 3D-printable, and open-source blimp platform.

The airship's inherent nonlinear dynamics and slow response time make it challenging to control. Moreover, its restrictive weight constraints inevitably lead to the need for controllers with low computational power requirements. In this regard, Spiking neural networks (SNNs) is a promising research direction, as they enable computing with highly parallel architectures made of simple integrate-and-fire neurons interconnected by weighted synapses. These architectures have demonstrated extremely low-power performance and are ideal for robotics applications [38][150]. Thus, on the one hand, they can be used for complex control tasks, since numerical simulations support their universal function approximation capabilities [55]. On the other hand, they provide low-power and energy efficiency traits [101]. Implementations of spiking flight neurocontrollers include a SNN for robust control of a simulated quadrotor in challenging wind conditions [78]. Although these SNN-based controllers for micro air vehicles (MAVs) have excelled in simulated environments, their main limitation is that they have not been evaluated in real-world experiments.

The scope of works that have implemented SNN controllers for MAVs in real scenarios is much more limited. The first work that integrates a SNN in the closed-loop control of a real-world flying robot is very recent [69]. There, the authors present a SNN for controlling the landing of a quadrotor by exploiting the optical flow divergence from a downward-looking camera. To address the learning problem of SNNs [167], they adopt an evolutionary training strategy. In [46], this SNN-based control system is further enhanced by using hardware specifically designed for neuromorphic applications.

The main contributions of this work are then twofold. First, we propose the design of an open-source, low-cost, lightweight blimp platform, which can be 3D printed, and allows for the inclusion of custom sensors and actuators. Second, we present an evolved altitude controller for our blimp platform based on a SNN, which relies solely on the sensory feedback provided by an airborne radar. This pushes forward the state-of-the-art on SNN controllers for real-world MAVs, taking as a basis the framework proposed in [69], by successfully demonstrating the performance of our radar-based neuromorphic controller on-board the blimp in real-world experiments, quantitatively comparing the results with those of an ANN and a proportional-integral-derivative (PID) controller.

1.1. Research question(s)

The central research question of this thesis is:

"To what extent does an altitude controller strategy for a blimp based on learning inspired spiking neural networks (SNNs) outperform a classical PID and an artificial neural network (ANN) controller, using only a short-range FMCW radar sensor?"

The previous research question can be divided into smaller subquestions. Each of these lower level questions provides an answer to part of the higher level one. All of them have been tackled throughout the project: either in the scientific paper (Part I), literature study (Part II) and/or methodologies report (Part III).

- What advantages and disadvantages does a controller based on learning inspired SNNs offer in contrast to a PID/ANN controller?
  - How much does the accuracy of a SNN based altitude controller for a blimp increase or decrease with respect to a PID/ANN controller?
  - How high is the control effort of the SNN controller strategy in contrast to the PID/ANN alternative?
  - What are the fundamental theoretical differences between an ANN and a SNN?
- What advantages does the use of a short-range FMCW radar sensor offer in comparison to vision-based sensors when used as inputs for the SNN controller?
  - How can the signals from the radar, as a frame-based sensor, be converted into spikes to be fed to the SNN in an efficient manner?
    - ◇ What type of data is produced by the sensor?
    - ◇ What are the most efficient algorithms to process these data?
    - ◇ Which activation functions should be chosen for each layer of the network and why?
  - Which type of decoding mechanisms exist to convert the spikes back into a real number (corresponding to the motor command for the blimp)?
- What advantages does an evolutionary training strategy for the ANN/SNN offer in contrast to other methods?
  - What is the theory behind the existing training methods for neural networks and how to they compare?
  - What are the learning challenges inherent to SNN learning and how to tackle them?
  - What alternatives exist to model the airship for the ANN/SNN training in simulation and which one fits best the purpose of this work?
- How much, qualitatively and quantitatively, does the reality gap impact the results and how to address it?
  - What state-of-the-art techniques have been used in the literature to minimize the effect of the reality gap?
  - Which techniques can be applied for the same purpose in this work and why?
- What improvements and future developments can be recommended?

1.2. Research objective

The main objective of this thesis is:

*”**To** develop an evolved neuromorphic altitude controller for a micro airship,*

***by** proposing an efficient platform design together with a suitable control algorithm based on learning inspired spiking neural networks,*

***using** only on-board processing, a short-range FMCW radar sensor,*

***and** quantitatively contrasting it with other classical control strategies.”*

In order to achieve this objective, several sub-goals can be formulated such that, the completion of each of these smaller objectives, will ultimately lead to the success of the final project. Concretely, in this case, three sub-goals are proposed:

The first sub-goal is to propose an efficient indoor blimp configuration by choosing appropriate lightweight, low-cost and computationally appropriate electronic components; combining them with an open-source, 3D-printable gondola design. The next sub-objective would be to develop an altitude control algorithm for the blimp by evolving the SNN controller in a simulated environment. Finally, the last sub-goal would be to evolve a classical ANN and PID controller, and quantitatively compare them with the previously developed SNN in a real-world environment.

1.3. Structure of this work

The main contributions of this master’s thesis are presented in the scientific paper in Part I, which can be read as a standalone document. This article primarily consists of, first, an introduction to the main concepts, relevant state-of-the-art, and main contributions; second, a description of the followed methodology, comprising the platform design, the chosen sensor, the proposed SNN and ANN architectures, neuron model, and evolutionary learning environment; third, a discussion of the obtained results; and finally, the corresponding concluding remarks based on these results. The remainder of this thesis provides support material for this paper. Therefore, readers unfamiliar with the topics of micro airships design and modelling, signal processing of radars, neural network evolution, and neuromorphic computing are encouraged to read this documentation beforehand.

Part II presents an in-depth review of the relevant literature on the topics of blimp design and modelling, remote sensing in UAVs with radars, blimp control strategies, neuromorphic computing and its learning strategies. Concretely, Chapter 2 introduces some of the state-of-the-art blimp design strategies, together with the airship’s dynamic model formulation and the existing airship simulations on the Gazebo framework. Next, Chapter 3 introduces the remote sensing strategies in UAVs, with special focus on the ones that are radar-based. It then delineates the main signal processing algorithm used to extract the range, velocity and radar information from the radar. Then, the main control strategies that have been implemented on blimps are explained in Chapter 4. Chapter 5 reviews the the field of neuromorphic computing by describing the main aspects of spiking and artificial neural networks, and contrasting them.

Finally, Part III documents an in-depth review of the followed methodology and some of the most relevant aspects that have been taken into account for the development of the final controllers proposed in this thesis. In Chapter 6 an introduction to the research methodology is presented. Then, a detailed description of the design process and the electronics involving the 3D gondola design, two assembly schematics, the selected electronic components and their integration, and the radar’s signal processing is presented in Chapter 7. The simulation framework utilized to evolve the controllers and the airship model used for the same purpose are explained in Chapter 8. Finally, Chapter 10 closes with the conclusions and some recommendations for future research.

I

Scientific paper

Evolved neuromorphic radar-based altitude controller for an autonomous open-source blimp

Marina González-Álvarez[1], Julien Dupeyroux[1], Federico Corradi[2], and Guido C.H.E. de Croon[1]

***Abstract*— Robotic airships offer significant advantages in terms of safety, mobility, and extended flight times. However, their highly restrictive weight constraints pose a major challenge regarding the available computational power to perform the required control tasks. Spiking neural networks (SNNs) are a promising research direction for addressing this problem. By mimicking the biological process for transferring information between neurons using spikes or impulses, they allow for low power consumption and asynchronous event-driven processing. In this paper, we propose an evolved altitude controller based on a SNN for a robotic airship which relies solely on the sensory feedback provided by an airborne radar. Starting from the design of a lightweight, low-cost, open-source airship, we also present a SNN-based controller architecture, an evolutionary framework for training the network in a simulated environment, and a control scheme for ameliorating the gap with reality. The system's performance is evaluated through real-world experiments, demonstrating the advantages of our approach by comparing it with an artificial neural network and a linear controller. The results show an accurate tracking of the altitude command with an efficient control effort.**

I. Introduction

The biological intelligence of living beings has long attracted us to explore their innate ability to learn complex tasks. For instance, despite their limitations in terms of cognitive capabilities and energy resources, flying insects can outperform some of the most advanced aerial robots nowadays on navigating autonomously through complex environments with fast and agile maneuvers [1]. In recent years, this inspiration has led to the development of controllers for unmanned aerial vehicles (UAVs) that mimic the structural and functional principles of the brain [2]. Neuromorphic control systems, as opposed to classical model-based approaches, can adapt to unknown scenarios (e.g. unmodelled dynamics or disturbances) due to their learning nature [3, 4]. Artificial neural networks (ANNs) [5] have proven successful for controlling different flying robots such as a hexacopter [6], a helicopter [7], or a quadrotor [8]. However, when it comes to light-weight micro air vehicles (MAVs), conventional ANNs present several disadvantages regarding energy consumption and response latency [9]. Spiking neural networks (SNNs), are a promising research direction in this regard. By processing information using just a small population of spikes with a precise relative timing, they allow for a more efficient learning and control [10, 11].

Among the main advantages of SNNs for aerial robotic applications, we can highlight that they enable computing with highly parallel architectures and provide low-power and energy efficiency traits [12, 13]. Additionally, they are universal value function approximators [14], which theoretically makes them suitable for addressing complex control tasks. However, they have not yet become a common method for designing controllers. This is mainly due to the discrete spiking nature of SNNs, which prevents the use of gradient-based optimization algorithms, such as the widely used back-propagation strategy for conventional ANNs. This makes the training process challenging. To tackle these issues, in this paper we present a SNN-based altitude controller for a low-cost micro airship with an open-source design, equipped with an airborne radar (Figure 1). The choice of the problem of tracking an altitude command is motivated by its relevance in key MAV applications such as autonomous package delivery [15, 16], or landing [17, 18], among others. On the other hand, the selection of a lighter-than-air craft as a test platform instead of a rotorcraft, is driven by the complementary advantages it poses, such as extended flight times, excellent ease of assembly, low acoustic footprint, low power consumption, and a simpler design [19]. By incorporating an airborne radar, the sensory feedback required for the control loop is robust to variant illumination and visibility conditions, while keeping the payload and computational requirements within reasonable limits.

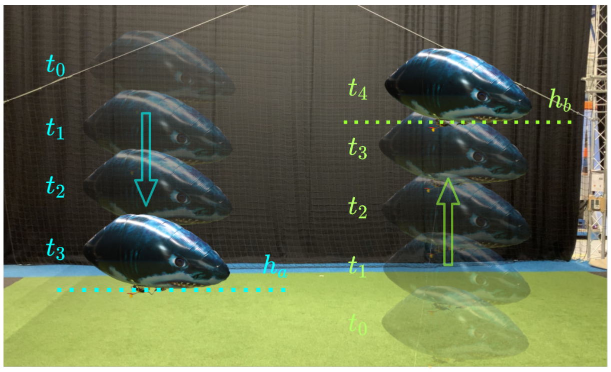


Fig. 1. Proposed autonomous altitude control system for an indoor airship. Evolution of the blimp altitude for two different setpoints (h_a and h_b) for several timestamps ($t_0, \ldots, t_4$).

[1] Micro Air Vehicle Laboratory, Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands. Contact: `j.j.g.dupeyroux@tudelft.nl`

[2] Ultra Low Power Systems for IoT, Stichting IMEC Nederland, Eindhoven, The Netherlands.

This work has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No. 826610. The JU receives support from the European Union's Horizon 2020 research and innovation program and Spain, Austria, Belgium, Czech Republic, France, Italy, Latvia, Netherlands. It has also received support from La Caixa Foundation (ID 100010434) under the code LCF/BQ/EU19/11710057.

The main contributions of this paper are then twofold. First, we present an evolved altitude controller for a micro air vehicle based on a SNN, which relies solely on the sensory feedback provided by an airborne radar. We successfully demonstrate the performance of the radar-based neurocontroller onboard the aerial platform in real-world experiments, quantitatively comparing the results with those of an ANN and a proportional-integral-derivative (PID) controllers. Second, we propose the design of an open-source, low-cost, lightweight blimp platform with a 3D printable gondola, that allows for the inclusion of custom sensors and actuators. This facilitates its replication and customization for different applications. The remainder of the paper is organized as follows: Section II provides an overview of the state-of-the-art in micro-airship design and spiking flight neurocontrollers. Afterwards, in Section III, we present the proposed MAV design, introduce the altitude control scheme based on the airborne radar, the structure of the SNN controller, the evolutionary strategy for training the network, and a blimp computational model to perform the training in a simulated environment. Then, in Section IV, we describe the real-world experimental setup as well as discuss the obtained results. Finally, Section V concludes the work and delineates future research directions.

II. Related Work

A. Micro-airship Design

Even though the golden era of giant cargo airships has faded, the advantages offered by lighter-than-air crafts prevail. Blimps are, slowly but surely, attracting increasing interest in the realm of unmanned aerial vehicles [20, 21]. They present endless possibilities in terms of their design. For example in [22], a three-propeller, low-cost platform is presented that is equipped with a camera and a compact, but closed-configuration gondola. An alternative design is proposed in [23], where the authors introduce a novel actuation mechanism based on two propellers mounted on a rotating shaft, which is oriented using a servomotor. Other examples of higher complexity include [24–26]. Although these alternatives have proven successful for their specific applications, they lack the versatility that can be achieved by leaving room for incorporating additional sensors and/or actuators. Besides, only [22] is open-source and lightweight enough to be mounted on commercially available blimp balloons. For the purpose of clear comparison, the main contributions of the state-of-the-art and our approach are summarized in Table I.

TABLE I

Comparison between the different blimp designs

Property	[22]	[23]	[24]	[25]	[26]	Ours
Easily customizable gondola	-	✓	-	-	-	✓
Low-cost design	✓	✓	✓	✓	-	✓
Open-source availability	✓	-	-	-	-	✓
Lightweight Microfoil blimp	✓	-	-	-	-	✓
Number of propellers	3	2	4	6	4	2
Number of servomotors	-	1	-	3	-	1

B. Spiking Neural Network-based MAV Control

The inherent nonlinear dynamics of most MAVs makes them challenging to control. Moreover, their restrictive weight constraints inevitably limit the computational power of the controller. SNNs enable computing with highly parallel architectures made of simple integrate-and-fire neurons interconnected by weighted synapses.

Implementations of spiking flight neurocontrollers include [27], where the authors propose a SNN for robust control of a simulated quadrotor in challenging wind conditions. They achieve a better performance in waypoint holding experiments compared with a hand-tuned PID and a multilayer perceptron network. Another example is presented in [28], where a SNN controller that adapts online to control the position and orientation of a flapping drone is proposed. SNNs have also been applied to obstacle avoidance tasks, as direct flight [14] or decision-making [29] controllers. In both cases they use reward-modulated learning rules for training the SNN. Although these MAV controllers have excelled in simulated environments, their main limitation is that they have not been evaluated in real-world experiments.

The scope of works that have implemented SNN controllers for MAVs in real scenarios is much more limited. The first work that integrates a SNN in the closed-loop control of a real-world flying robot is very recent [30]. There, the authors present a SNN for controlling the landing of a quadrotor by exploiting the optical flow divergence from a downward-looking camera and the readings of an inertial measurement unit (IMU). To address the learning problem of SNNs [31], they adopt an evolutionary training strategy. In [32], this controller is enhanced by using hardware specifically designed for neuromorphic applications. Although not tested in free flight experiments, the potential advantages of SNN controllers implemented in these devices are also demonstrated in [33].

Our work aims to extend the framework proposed in [30], by (1) controlling the altitude instead of landing; (2) considering an open-source micro blimp, which has less control authority and harder to model dynamics than a quadrotor; and (3) exploiting solely the range measurements provided by a radar, reducing the number of required sensors on-board (i.e., no IMU).

III. Methodology

A. Open-source Micro-airship

The proposed design for the micro autonomous airship is illustrated in Figure 2. The reader interested in replicating the platform can find further details, links to re-sellers, prices, and parts for 3D printing at: `https://github.com/tudelft/blimp_snn`. The airship's gondola can be 3D printed and assembled in a modular fashion, with a total frame weight of just 9g. Due to its open configuration, the components mounted on the gondola can be easily interchanged, leaving room for versatility on the selection of sensors and actuators. In addition, we include a rotary shaft with a case for accommodating the propellers on both

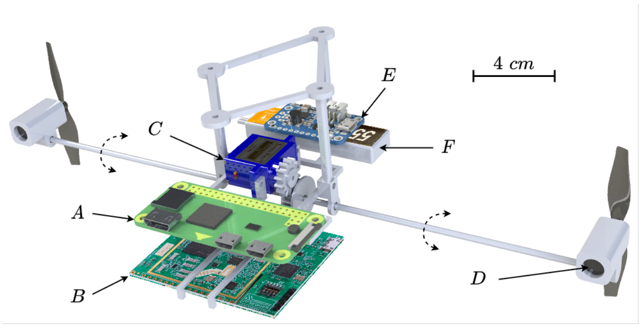


Fig. 2. Scheme of the proposed airship design. (A) Raspberry Pi W Zero; (B) 24 GHz Infineon Radar Position2Go; (C) Sub-micro Servo SG51R; (D) 8520 Coreless Motor; (E) PowerBoost 500 Basic; (F) 550mA 3.8V Li-Po Battery.

ends for controlling the altitude. Finally, we incorporate four hitches on top of the gondola, where we tape Velcro strips for attaching the envelope. Regarding the electronic components, we use a Raspberry Pi W Zero as the central communication and control unit, running the Raspbian Lite operating system. The robot's steering is achieved through the micro servomotor mounted on the gondola and the two core-less direct current (DC) motors attached at each end of the shaft. Specifically, the servo is responsible for the rotation of the shaft, up to 180°, and the DC motors allow for an independent control of the thrust on each side. Additional peripheral components include a step-up voltage regulator, a 500 mAh Li-Po battery and a motor driver. Finally, a fast chirp frequency-modulated continuous wave (FMCW) radar module from Infineon with a resolution of ±20 cm is used as a ranging sensor for the closed-loop control. Concerning the airship's envelope, the material chosen is Microfoil due to its excellent gas retention capabilities [22]. We select a model that provides the largest achievable payload among the commercially available miniature blimps (150g) while keeping a relatively low price. For our application, we use helium as the lifting gas. Considering all the aforementioned elements, the proposed platform weights a total of 147g. To integrate the different components and perform the computations on-board we adopt the Robot Operating System (ROS) [34] framework. In addition, a tele-operation package to manually control the airship from a ground computer keyboard via a secure shell (SSH) connection is also provided in the repository included at the beginning of this section.

B. Altitude Controllers

In order to control the autonomous airship's altitude, the commands are provided in terms of motor voltages, $u \in [-u_{\max}, u_{\max}]$ $[V]$, with $u_{max} = 3.3$ $[V]$. The larger the absolute value of u, the more thrust the propellers provide, and therefore, the greater the acceleration of the blimp will be. The sign of the voltage does not represent the polarity of the electric signal, but the direction in which the airship is moving. Thus, when $u > 0$, the robot moves upwards and, when $u < 0$, the robot moves downwards, with the shaft rotated 180°.

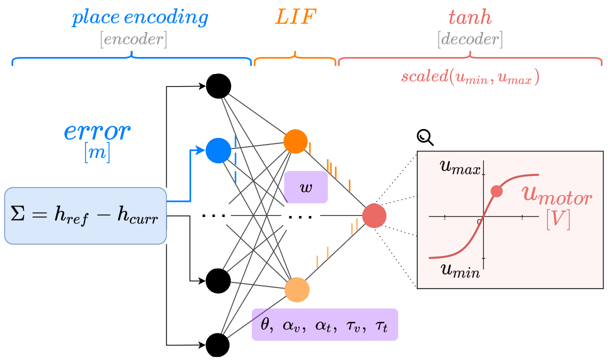

place encoding
[encoder]
LIF
tanh
[decoder]
scaled(u_{min}, u_{max})
error
[m]
$\Sigma = h_{ref} - h_{curr}$
w
u_{max}
u_{motor}
[V]
u_{min}
θ, α_v, α_t, τ_v, τ_t

Fig. 3. Scheme of the SNN controller architecture. The (evolved) network parameters are highlighted in violet, being w the synaptic weights, θ the spiking threshold, $\alpha_{v/t}$ the scaling constant for the increase of the voltage/trace by a single spike, and $\tau_{v/t}$ the decay for the voltage/trace.

To determine the required control actions for tracking an arbitrary reference altitude, h_{ref}, we process the readings from the Infineon Position2Go airborne radar to get an estimate of the current height of the blimp, h_{curr} [35]. Specifically, the range-Doppler algorithm [36] is used for the processing.

Afterward, a median filter and a moving average filter are used to decrease the signal noise and remove possible outliers. Then, to effectively track an arbitrary altitude command, we design a controller that provides a mapping between the altitude error, $h_{ref} - h_{curr}$, and the motor voltages, u, such that the former is minimized. We consider three distinct approaches for benchmarking purposes: a linear PID, an artificial neural network, and a spiking neural network.

1) Proportional-integral derivative controller: A conventional PID is one of the most simple, yet widespread methods for addressing control problems. In discrete form, the mapping between the error signal $e_k = h_{ref}(k) - h_{curr}(k)$ and the motor command u_k is given by [37]:

$$u_k = K_p e_k + \frac{K_d}{T}\left(e_k - e_{k-1}\right) + K_i T\left(e_k + e_{k-1}\right) \qquad (1)$$

where K_p, K_i and K_d refer to the proportional, integral and derivative gains, respectively, and T to the sampling period. These are tuned empirically using the proposed MAV platform until we achieve the desired behavior.

2) Artificial neural network controller: For the ANN case, the tracking error $h_{ref} - h_{curr}$ is directly fed into the network in the form of a continuous signal. The proposed neuron architecture, from the input to the output layer, follows a $1-3-2-1$ scheme. The input and the two hidden layers operate with a $\tanh()$ activation function. At the output layer, a linear neuron provides the value of the motor command u, clamped to the interval $\pm u_{\max}$ $[V]$.

3) Spiking neural network controller: The proposed SNN architecture is illustrated in Figure 3. The network consists of three fully connected layers of sizes 10, 5 and 1 neuron, from the input to the output. The input layer acts as a position placeholder that encodes the altitude error signal into spikes. More specifically, the input values of $h_{ref} - h_{curr}$ are divided into 10 intervals, with each of them assigned to a different neuron. The range of the first and last intervals corresponds to $]-\infty, -0.4[$ and $]0.4, \infty[$, respectively, while the remainder 8 are uniformly distributed between $[-0.4, 0.4]$. Each time the altitude error falls within one of these "gaps", the corresponding neuron fires a single spike. The hidden layer consists of five leaky integrate-and-fire (LIF) neurons, where the membrane potential of the i-th neuron, $v_i(t)$, is governed by the following equation:

$$v_i(t) = \tau_{v_i} \cdot v_i(t - \Delta t) + \alpha_{v_i} u_i(t) \qquad i = 1, \ldots, 5 \tag{2}$$

referring $\tau_{v_i} \in [0, 1]$ to the decay factor per time-step Δt, α_{v_i} to a scaling constant, and $u_i(t)$ to the synaptic input current:

$$u_i(t) = \sum_{j=1}^{10} w_{ij} s_j(t). \tag{3}$$

that is, multiplying the incoming spikes from the j-th input neuron $s_j(t)$, by the synaptic weights w_{ij}. Whenever the membrane potential $v_i(t)$, reaches a certain threshold θ_i, a postsynaptic spike is triggered and $v_i(t)$ resets back to 0. The output layer decodes the spikes back into a real value. It consists of a single non-spiking neuron with a scaled tanh() activation function. The neuron conducts a weighted sum of the so-called spike traces, $X_i(t)$, which is computed as:

$$X_i(t) = \tau_{t_i} \cdot X_i(t - \Delta t) + \alpha_{t_i} s_i(t), \tag{4}$$

being the definition of τ_{t_i} and α_{t_i} analogous to τ_{v_i} and α_{v_i}. The resulting value is scaled within the control limits, $\pm u_{\max}$. Following this, the motor command, u, is given by:

$$u(t) = u_{\max} \cdot \tanh\left(\sum_{i=1}^{5} w_i X_i(t)\right) \tag{5}$$

C. Evolutionary Framework

For training the neural network controllers we adopt an evolutionary strategy. Each evolution begins with a randomly initialized population of N individuals. As in [30], a mutation-only procedure is then followed. The offspring is obtained by performing a randomized tournament selection of M individuals i.e. randomly selecting M aspirants from the population and keeping the one with the best fitness. This is repeated N times, so that the population size is invariant. The n-th individual is mutated with a probability of $p_{mut}^{(n)} = 0.4$, and its m-th parameter with $p_{mut}^{(m)} = 0.6$. These mutations take place according to uniform probability distributions $\mathscr{U}\{,\}$, whose range is shown in Table II and III for the SNN and ANN, respectively. For the latter, the open parameters are the biases, b_i, and analogously to SNNs, the weights, w_{ij}.

TABLE II
SNN PARAMETERS MUTATED DURING EVOLUTION

Parameter	Domain	Mutation
w_{ij}	$[-5,\ldots,5]$	$\mathscr{U}\{-2.5,2.5\}$
θ_i	$[0,\ldots,1]$	$\mathscr{U}\{-0.5,0.5\}$
α_{v_i/t_i}	$[0,\ldots,2]$	$\mathscr{U}\{-1.0,1.0\}$
τ_{v_i/t_i}	$[0,\ldots,1]$	$\mathscr{U}\{-0.5,0.5\}$

TABLE III
ANN PARAMETERS MUTATED DURING EVOLUTION

Parameter	Domain	Mutation
w_{ij}	$[-5,\ldots,5]$	$\mathscr{U}\{-2.5,2.5\}$
b_i	$[-5,\ldots,5]$	$\mathscr{U}\{-2.5,2.5\}$

The mutated offspring is then evaluated in a model-based simulation environment (see Section III-D), where a source of random Gaussian noise is added to the radar signal. Since this randomization stimulates the persistence of controllers that are independent of such disturbances, it helps minimizing the reality gap [38]. During the evaluation, a set of 10 different reference altitudes $h_{ref} \in [0, 3]$ is provided along a total simulated duration of $T = 15$ seconds each. The fitness of each individual is then quantified as the root mean squared altitude error (RMSAE):

$$\text{RMSAE} = \sqrt{\frac{1}{T}\sum_{k=0}^{T}\left(h_{ref}(k) - h_{curr}(k)\right)^2} \tag{6}$$

During the evolution process, a *hall of fame* which holds the 5 best performing individuals across all generations, is maintained. This prevents discarding those who have achieved a good performance. After N_{gen} generations, the individuals are also reevaluated on five more random sets of altitudes to increase the robustness. The best-performing ones are selected for further real-world experiments.

D. Model-based Simulation Environment

The altitude controllers evolve in a simulated environment since it would be infeasible to perform all the required evaluations in the real world. For that, we develop a dynamical model of the blimp. Essentially, the idea is to obtain a mapping between the motor commands provided by the controller and the evolution of the blimp's altitude over time. We assume that the acceleration at the k-th time step $\ddot{h}_k$, is proportional to the voltage applied to the motors, u, i.e.

$$\ddot{h}_k = a_1 u_{k-1} + a_2 u_{k-2} \tag{7}$$

where a_i is the proportionality constant for the motor command at time instant $k - i$. However, since the acceleration cannot be directly measured with the radar sensor, we can instead express this relation in terms of the measured altitude, h by taking Euler's discretization of the derivative

$$h_k - 2h_{k-1} + h_{k-2} = a_1 u_{k-1} + a_2 u_{k-2} \tag{8}$$

Applying the Z-transform, we obtain the following transfer function, which maps the commands u_k to the altitude h_k, and allows us to easily simulate the blimp's dynamic behavior

$$h_k = \frac{a_1 z^{-1} + a_2 z^{-2}}{1 - 2z^{-1} + z^{-2}} u_k \tag{9}$$

To determine the unknown parameters a_i, we collected a dataset by tele-operating the blimp and measuring its altitude over time. After subtracting the mean, we infer the model

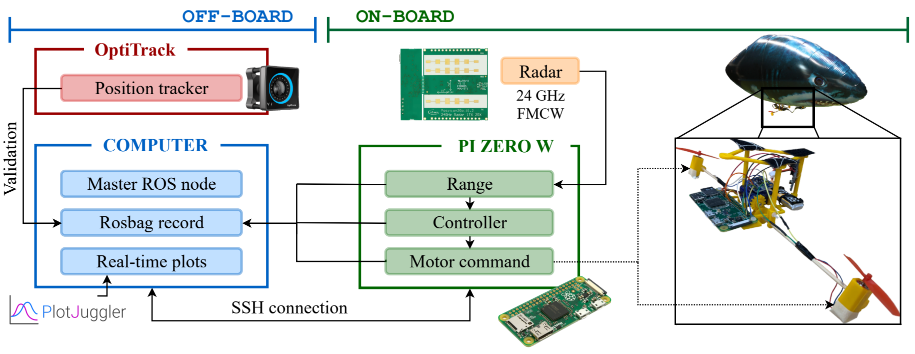


Fig. 4. System overview. After processing the feedback provided by the radar sensor, an estimate of the range is sent to the Raspberry Pi Zero W control unit. Only data recording and real-time plotting operations are conducted on the ground computer, which communicates with the Pi via an SSH connection. The OptiTrack is used during the post-processing stage just for validation purposes.

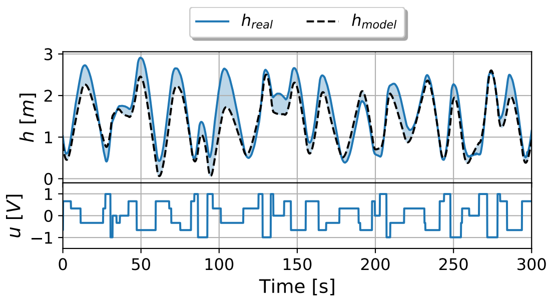


Fig. 5. Validation of the blimp model. ***Bottom:*** Motor commands. ***Top:*** The ground truth evolution of the altitude, h_{real}, compared with the evolution predicted by the model, h_{model}. The error is represented by the blue area.

parameters by minimizing the normalized root mean squared altitude error (NRMSAE):

$$\text{NRMSAE} = \sqrt{\frac{\sum_k \left(\hat{h}_k - h_k\right)^2}{\sum_k \hat{h}_k^2}} \tag{10}$$

which can be interpreted as a measure of how well the expected response h_k matches the observed data $\hat{h}_k$.

IV. Results

A. Experimental Setup

1) Simulation: To train the neural controllers, we evolved five randomly initialized populations of 100 individuals through 300 generations, following the procedure described in Section III-C. The implementation of the evolutionary optimization is based on the Distributed Evolutionary Algorithms in Python (DEAP) [39] framework, while the simulation of the networks is performed by means of the PySNN library [40].

2) Real-World: An overview of the setup is shown in Figure 4. The on-board control unit is a 1GHz single-core processor Raspberry Pi Zero W with 512MB RAM. The Infineon Position2Go radar provides altitude measurements. The control loop runs at a rate of 5 Hz.

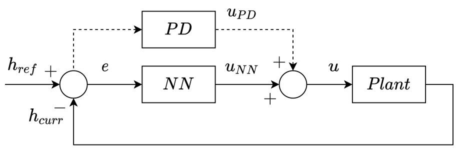


Fig. 6. Proposed control scheme for closing the reality gap of the neural network-based controllers trained in simulation.

B. Blimp Model

Following the procedure explained in Section III-D, we infer the parameters of a blimp model of the form (9), based on experimental data gathered using the real hardware. The resulting model is given by:

$$h_{model} = 10^{-3} \cdot \frac{-0.969z^{-1} + 1.019z^{-2}}{1 - 1.99z^{-1} + 0.99z^{-2}} u_{motor} \tag{11}$$

where we have also considered the denominator's parameters as open, yielding almost identical values to the theoretical ones. In Figure 5 we show a comparison between the evolution of the altitude predicted by the model and the ground truth when applying identical motor commands. We can see that we are able to reproduce the blimp's behavior using the proposed data-driven model, with a RMSAE of 0.27m over the 300 seconds run.

C. Controller Evaluation

We evaluate the performance of three different altitude controllers based on a linear PID, an ANN, and a SNN. The tracking precision is tested on a sequence of five different waypoints $h_d = \{3, 2, 1, 2.5, 1.5\}$m, maintained during 60s. Additionally, due to the existing mismatch between the linear simulated model and the real robot, directly taking the output of the evolved networks as the rotor commands would lead to deficient performance. To reduce the reality gap, we propose the control scheme shown in Figure 6. Essentially, we tune a parallel PD controller in the real-world setup to account for the contribution of disturbances and neglected dynamics. The chosen gains are small so that the PD addition is kept at a maximum of 16% with respect to the magnitude of the motor command.

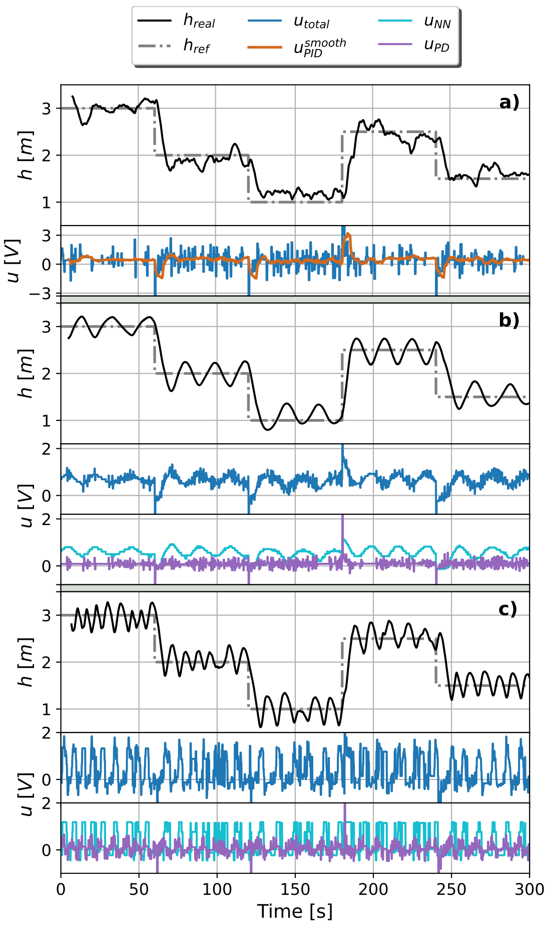


Fig. 7. Experimental evaluation of the considered controllers. For all three sub-figures, at the bottom we have the motor commands, and on top, the evolution of the blimp's altitude h_{real} compared with the reference h_{ref}. ***(a) PID*:** u refers to the motor command, and u_{smooth} is obtained after smoothing it with a moving average. ***(b) ANN*:** u_{ANN} stands for the output of the evolved controller, u_{PD} to the contribution of the PD parallel controller, and $u_{total} = u_{ANN} + u_{PD}$. ***(c) SNN*:** Analogous to the ANN controller.

1) PID controller: The experimental results are depicted in Figure 7(a), using the gains K_p, K_i and K_d indicated in Table IV. We can see that we can track the altitude commands effectively. Quantitatively, we obtain a RMSAE of 0.29m, which indicates a satisfactory performance, considering that the uncertainty of the radar sensor is of $\pm$0.2m.

2) ANN controller: The obtained results are shown in Figure 7(b), after reducing the reality gap with the PD gains specified in Table IV. We can see that the blimp effectively converges to the altitude set-point but presents an oscillatory behavior. This is mainly because of two reasons: the minor contribution of the diminished discrepancies between the model and the vehicle's inherent dynamics; and the slow responsiveness of the system, especially when the motor commands are not too abrupt, as it is the case. However, it can be noted that the trajectory is smoother than with a PID. The RMSAE now corresponds to 0.27m.

3) SNN controller: The experimental results for this case are displayed in Figure 7(c), using the PD gains from Table IV. We can observe that the behavior and performance are similar to the previous case, but with faster oscillations, due to the output's binary nature caused by the presence or absence of the spike. The RMSAE is also of 0.27m.

TABLE IV
GAINS COMPARISON BETWEEN THE DIFFERENT CONTROLLERS

Strategy / Gain	**PID**	**ANN**	**SNN**
K_p	6.0	1.3	1.4
K_i	0.4	-	-
K_d	0.9	0.4	0.3

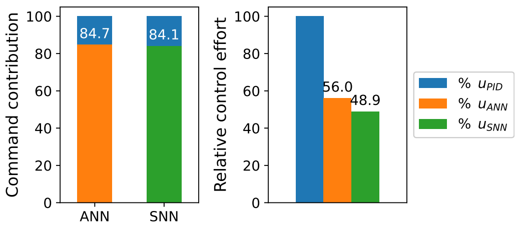


Fig. 8. Comparison between the PID, ANN and SNN controllers. ***Left:*** Motor command contribution of the neurocontrollers against the parallel PD. ***Right:*** Relative control effort, computed according to Equation (12).

4) Comparison: Figure 8 shows a comparison between the control commands provided by each controller. On the left, we observe that the contribution of the parallel PD is similar for both neuromorphic controllers, remaining below 16%. On the right, we perform the analysis in terms of the magnitude of the control effort, relative to that of the PID,

$$\%u_{ANN/SNN} = \frac{\sum_k |u_{ANN/SNN}(k)|}{\sum_k |u_{PID}(k)|} \cdot 100\% \qquad (12)$$

which gives an estimate of how energy efficient the controllers are. We can see that, even though the three control strategies present a similar RMSAE, the neurocontrollers, and especially our SNN design, exhibit less control effort, which saves energy.

V. Conclusion

Despite the recent advancements, it is still challenging for micro air vehicles (MAVs) to carry on-board the complex controllers required to fly autonomously. In this paper, we push the state-of-the-art in MAV control by presenting a novel altitude controller based on a spiking neural network (SNN). Our SNN architecture is evolved within a model-based simulation. The results obtained in real-world experiments successfully demonstrate the system's performance. By comparing it with a standard PID and an artificial neural network, we corroborate the advantages offered by SNNs in terms of adaptability and low control effort. Although there is still room for improvement in terms of performance and network complexity, future research will involve the use of specific neuromorphic hardware to better reflect the promises of neuromorphic computing in MAV control.

References

[1] Z. Zheng, J. S. Lauritzen, E. Perlman, C. G. Robinson, M. Nichols, D. Milkie, O. Torrens, J. Price, C. B. Fisher, N. Sharifi, S. A. Calle-Schuler, L. Kmecova, I. J. Ali, B. Karsh, E. T. Trautman, J. A. Bogovic, P. Hanslovsky, G. S. Jefferis, M. Kazhdan, K. Khairy, S. Saalfeld, R. D. Fetter, and D. D. Bock, "A complete electron microscopy volume of the brain of adult drosophila melanogaster," *Cell*, vol. 174, no. 3, pp. 730–743.e22, 2018.

[2] Y. Jiang, C. Yang, J. Na, G. Li, Y. Li, and J. Zhong, "A brief review of neural networks based learning and control and their applications for robots," *Complexity*, Oct 2017.

[3] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008, robotics and Neuroscience.

[4] J. Yu, M. Tan, J. Chen, and J. Zhang, "A survey on cpg-inspired control models and system implementation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 441–456, 2014.

[5] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, 2018.

[6] B. Kusumoputro, H. Suprijono, M. A. Heryanto, and B. Y. Suprapto, "Development of an attitude control system of a heavy-lift hexacopter using elman recurrent neural networks," in *2016 22nd International Conference on Automation and Computing (ICAC)*, 2016, pp. 27–31.

[7] H. Suprijono and B. Putro, "Direct inverse control based on neural network for unmanned small helicopter attitude and altitude control," *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, pp. 99–102, 2017.

[8] M. Heryanto, H. Suprijono, B. Suprapto, and B. Putro, "Attitude and altitude control of a quadcopter using neural network based direct inverse control scheme," *Advanced Science Letters*, vol. 23, no. 5, pp. 4060–4064, May 2017.

[9] A. Yousefzadeh, S. Hosseini, P. Holanda, S. Leroux, T. Werner, T. Serrano-Gotarredona, B. L. Barranco, B. Dhoedt, and P. Simoens, "Conversion of synchronous artificial neural network to asynchronous spiking neural network using sigma-delta quantization," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2019, pp. 81–85.

[10] W. Maas, "Networks of spiking neurons: The third generation of neural network models," *Trans. Soc. Comput. Simul. Int.*, vol. 14, no. 4, p. 1659–1671, Dec. 1997.

[11] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in Neuroscience*, vol. 10, p. 508, 2016.

[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, no. 1, pp. 82–99, 2018.

[13] J. Stuijt, M. Sifalakis, A. Yousefzadeh, and F. Corradi, "μBrain: An Event-Driven and Fully Synthesizable Architecture for Spiking Neural Networks," *Frontiers in Neuroscience*, vol. 15, p. 538, 2021.

[14] G. Foderaro, C. Henriquez, and S. Ferrari, "Indirect training of a spiking neural network for flight control via spike-timing-dependent synaptic plasticity," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 911–917.

[15] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

[16] B. Arbanas, A. Ivanovic, M. Car, T. Haus, M. Orsag, T. Petrovic, and S. Bogdan, "Aerial-ground robotic system for autonomous delivery tasks," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5463–5468.

[17] G. de Croon, H. Ho, C. D. Wagter, E. van Kampen, B. Remes, and Q. Chu, "Optic-flow based slope estimation for autonomous landing," *International Journal of Micro Air Vehicles*, vol. 5, no. 4, pp. 287–297, 2013.

[18] A. Borowczyk, D.-T. Nguyen, A. Phu-Van Nguyen, D. Q. Nguyen, D. Saussié, and J. L. Ny, "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle**this work was partially supported by cfi jelf award 32848 and a hardware donation from dji." *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 10 488–10 494, 2017, 20th IFAC World Congress.

[19] Y. Li, M. Nahon, and I. Sharf, "Airship dynamics modeling: A literature review," *Progress in Aerospace Sciences*, vol. 47, no. 3, pp. 217 – 239, 2011.

[20] P. G. Artaxo, A. Bourgois, H. Sardinha, H. Vieira, E. C. de Paiva, A. R. Fioravanti, and P. A. Vargas, "Autonomous cooperative flight control for airship swarms," 2020.

[21] E. Price, Y. T. Liu, M. J. Black, and A. Ahmad, "Simulation and control of deformable autonomous airships in turbulent wind," 2020.

[22] G. Gorjup and M. Liarokapis, "A low-cost, open-source, robotic airship for education and research," *IEEE Access*, vol. 8, pp. 70 713–70 721, 2020.

[23] S. U. Ferdous, A. Mohammadi, and S. Lakshmanan, "Developing a low-cost autonomous blimp with a reduced number of actuators," in *Unmanned Systems Technology XXI*, C. M. Shoemaker, H. G. Nguyen, and P. L. Muench, Eds., vol. 11021, International Society for Optics and Photonics. SPIE, 2019, pp. 73 – 80.

[24] K. Watanabe, N. Okamura, and I. Nagai, "Closed-loop control experiments for a blimp robot consisting of four-divided envelopes," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, 2015, pp. 2568–2573.

[25] S. Oh, S. Kang, K. Lee, S. Ahn, and E. Kim, "Flying display: Autonomous blimp with real-time visual tracking and image projection," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 131–136.

[26] M. Burri, L. Gasser, M. Käch, M. Krebs, S. Laube, A. Ledergerber, D. Meier, R. Michaud, L. Mosimann, L. Müri, C. Ruch, A. Schaffner, N. Vuilliomenet, J. Weichart, K. Rudin, S. Leutenegger, J. Alonso-Mora, R. Siegwart, and P. Beardsley, "Design and control of a spherical omnidirectional blimp," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1873–1879.

[27] D. Howard and A. Elfes, "Evolving spiking networks for turbulence-tolerant quadrotor control," *Artificial Life Conference Proceedings*, no. 26, pp. 431–438, 2014.

[28] T. S. Clawson, S. Ferrari, S. B. Fuller, and R. J. Wood, "Spiking neural network (snn) control of a flapping insect-scale robot," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 3381–3388.

[29] F. Zhao, Y. Zeng, and B. Xu, "A brain-inspired decision-making spiking neural network and its application in unmanned aerial vehicle," *Frontiers in Neurorobotics*, vol. 12, p. 56, 2018.

[30] J. J. Hagenaars, F. Paredes-Valles, S. M. Bohte, and G. C. H. E. de Croon, "Evolved neuromorphic control for high speed divergence-based landings of mavs," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, p. 6239–6246, Oct 2020.

[31] X. Wang, X. Lin, and X. Dang, "Supervised learning in spiking neural networks: A review of algorithms and evaluations," *Neural Networks*, vol. 125, pp. 258–280, 2020.

[32] J. Dupeyroux, J. J. Hagenaars, F. Paredes-Vallés, and G. de Croon, "Neuromorphic control for optic-flow-based landings of mavs using the loihi processor," *CoRR*, vol. abs/2011.00534, 2020.

[33] A. Vitale, A. Renner, C. Nauer, D. Scaramuzza, and Y. Sandamirskaya, "Event-driven vision and control for uavs on a neuromorphic chip," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[34] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[35] N. Wessendorp, R. Dinaux, J. Dupeyroux, and G. de Croon, "Obstacle avoidance onboard mavs using a fmcw radar," 2021.

[36] V. Winkler, "Range doppler detection for automotive fmcw radars," *2007 European Radar Conference*, pp. 166–169, 2007.

[37] K. Ogata, *Discrete-Time Control Systems*. USA: Prentice-Hall, Inc., 1987.

[38] K. Y. Scheper and G. C. de Croon, "Evolution of robust high speed optical-flow-based landing for autonomous mavs," *Robotics and Autonomous Systems*, vol. 124, p. 103380, 2020.

[39] F.-A. Fortin, F.-M. D. Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "Deap: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, no. 70, pp. 2171–2175, 2012.

[40] B. Büller, "Pysnn," https://github.com/BasBuller/PySNN, 2019.

II

Related work

2 Blimp design and modelling overview

Even though the golden era of giant cargo airships has faded, the advantages offered by lighter-than-air (LTA) crafts prevail. Their high mobility and less restrictive path planning constraints make them a better alternative than ground robots in terms of indoor navigation and exploration. Furthermore, their great autonomy, ease of assembly, low acoustic footprint, low power consumption and, consequently, general low costs, also pose complementary advantages to those that rotorcraft have to offer, allowing them to compete face-to-face with the latter in terms of small UAVs navigation.

In order to continue with the successive chapters, it is first necessary to build a clear understanding of the existing platforms, developed in recent years, and their mechanical properties and design. This review can be found in Section 7. Of course, this analysis would be empty without the proper mathematical formulation supporting the choice for a certain geometric configuration and delineating the way airships move in our surroundings. Therefore, this is presented in Section 2.2.

2.1. Autonomous blimp design strategies

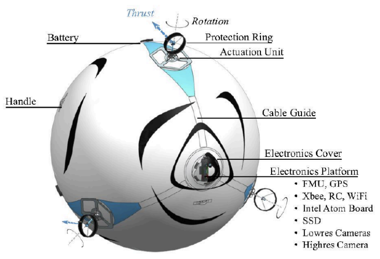
Table 2.1 features an overview of the different types of autonomous airship platforms available in the literature that are most interesting for this study based on their size, number of actuators and geometrical configuration. First, a small comment will be made on the most relevant features of each of the non-highlighted proposals. Second, the two configurations highlighted in bold will be analysed in more detail and compared, because of their compelling particular properties and suitability for the present work.

Contributors	**# motors**
[49], [135]	2 DC + 1 servo
[66], [3]	3 DC
[29], [171]	4 DC
[121]	6 servo

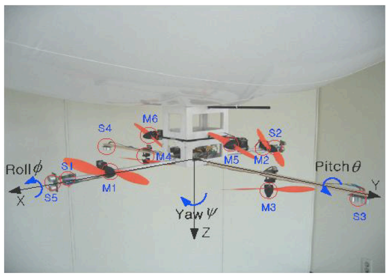
Table 2.1: An overview of the various types of autonomous blimp designs in the literature

On the other hand, the design of a long oval-shaped outdoor airship with a rotational stereo camera and laser range-finder aimed at gathering information after large-scale disasters to facilitate rescue missions is proposed in [135]. In [171], a new body configuration consisting on four partial envelopes centering on a gondola and four propellers is presented. A novel scheme consisting of three pairs of light-weight servomotors located on each of the x-y-z axis, so that each pair can control both the translation in the corresponding direction and also the pitch, yaw and roll rotations is suggested in [121]. *Air Shark*, in [1], is a fish-shaped blimp in which the center of gravity can be displaced forward or backward to control the pitch and the tail movement makes the airship move through the air. Figure 2.1 graphically shows the distinct examples presented up to this point.

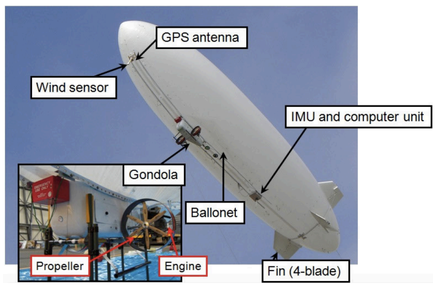
(a) [3]



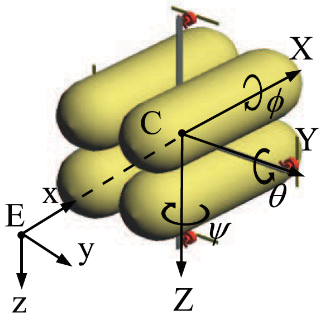
(b) [29]



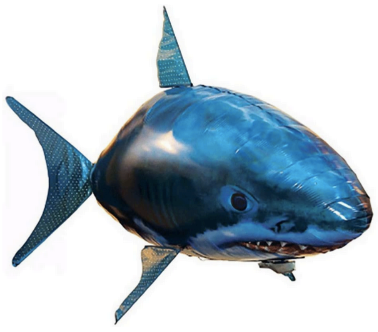
(c) [121]



(d) [135]



(e) [171]

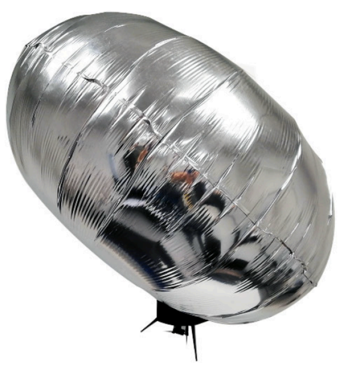


(f) [1]

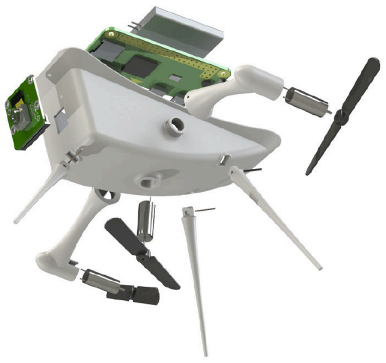
Figure 2.1: Visual representation of several blimp model examples in Table 2.1.

As mentioned at the beginning of this section, special focus will be put on the works by Ul Ferdous [49] and Gorjup [66], which correspond to the highlighted publications in Table 2.1:

1. First, the design of a three-propeller low-cost and open-source indoor airship equipped with a camera for education and research (see Figure 2.2a) is presented in [66]. All a list with each of the utilized electronic components and their vendors, the developed ROS interface to control the speed of the motors and the gondola computational designs in SolidWorks (see Figure 2.2b) are accessible online[1] to everyone to be directly 3D printed. Moreover, a study analysing the helium permeability and mechanical properties of several envelopes material was conducted. Finally, two different gondola positions were considered in two proof-of-concept flight stability experiments. It is worth to highlight the comprehensiveness and accessibility inherent to this proposal.



(a) General configuration.



(b) Assembly concept design.

Figure 2.2: Indoor airship model as proposed in [66].

2. Second, Ul Ferdous et al. [49] introduce a novel actuation mechanism based on only one servo-motor and two DC motors to increase the blimp's operating flight time by reducing the weight of its components and choosing a simpler, more light-weight gondola design. The notable weight

[1]http://www.newdexterity.org/openairship

restrictions that small indoor airships pose, together with the promising results achieved by this arrangement make it a very appealing option that may serve as a basis for the current work. Figure 2.3 exhibits this platform's assembly configuration. The shaft is able to rotate from 0° up to 180° by means of two 3D printed gears connected to the servomotor, also allowing the blimp to perform complex maneuvres.

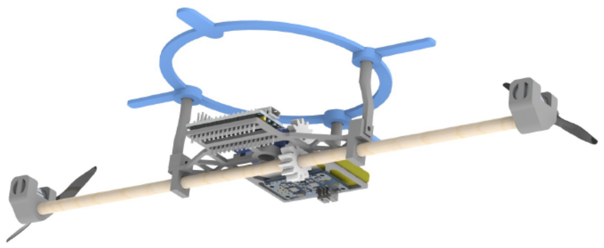


Figure 2.3: Assembly concept in [49].

2.2. Blimp dynamic model formulation

To be able to perform simulations of the blimp, optimize its design, test different control strategies and, ultimately, study its behavior, a proper mathematical model is needed. Therefore, in this section, the underlying mathematical formulation of the dynamic model of blimp is presented, based on [39] and [178]. A very detailed literature review on the matter can also be found in [102].

The dynamic model of the blimp describes its full nonlinear six degrees of freedom (DoFs) motion. Therefore, it relates the moments and forces acting on the platform as well as its acceleration, in six DoFs. Before diving into the actual equations, a few assumptions have to be made [39]:

1. Aerolastic effects can be ignored, as the platform is assumed to form a rigid body.
2. The center of volume and gravity, C_v and C_g, both lie in the plane of symmetry, since the blimp is assumed to be symmetric about the v_x and v_z plane.
3. The fact that the gondola, attached right under the hull, contains most of the equipment for power, actuation and sensing, allows to make the assumption that the center of gravity C_g lies right below the center of buoyancy C_b. This implies that the airship is stabilized about the pitch and roll axes.

With this assumptions in mind, the dynamic model is expressed as:

$$\boldsymbol{M}\dot{\boldsymbol{v}} = \boldsymbol{F_c} + \boldsymbol{F_g} + \boldsymbol{F_a} + \boldsymbol{F_p}, \tag{2.1}$$

where $\boldsymbol{M}$ accounts for the mass and inertia matrix, $\boldsymbol{F_c}$ corresponds to the force vector resulting from the Coriolis effect, $\boldsymbol{F_g}$ comprises the gravitational and buoyancy induced forces and moments, $\boldsymbol{F_a}$ accounts for the aerodynamic forces and moments and, finally, $\boldsymbol{F_p}$ is the vector of propulsion forces and moments that arise from the propeller thrust. In the upcoming subsections, the different elements of Equation 2.1 are analyzed in more detail.

Mass and inertia

The total mass and inertia matrix arises from adding the the rigid-body inertia matrix and the added-mass effects matrix, which accounts for the resistance generated by the large amount of particles that are displaced when the blimp moves:

$$\boldsymbol{M} = \boldsymbol{M_{RB}} + \boldsymbol{M_A} = \begin{bmatrix} m'_x & 0 & 0 & 0 & 0 & 0 \\ 0 & m'_y & 0 & 0 & 0 & 0 \\ 0 & 0 & m'_z & 0 & 0 & 0 \\ 0 & 0 & 0 & I'_x & 0 & 0 \\ 0 & 0 & 0 & 0 & I'_y & 0 \\ 0 & 0 & 0 & 0 & 0 & I'_z \end{bmatrix}, \tag{2.2}$$

where $\boldsymbol{M_{RM}} = diag(m, m, m, I_x, I_y, I_z)$ and $\boldsymbol{M_A} = diag(m_{A_x}, m_{A_y}, m_{A_z}, I_{A_x}, I_{A_y}, I_{A_z})$. For more details on how the added mass terms are calculated, please refer to [39].

Coriolis effect
Whenever a fictitious force is exerted on a moving body when the reference frame is not inertial, the Coriolis effect appears. Concretely, this effect occurs whenever a motion can be expressed as $\omega \times v$; this is, when it is composed of linear and rotational velocities. The Coriolis force tends to preserve the initial direction of motion regardless of the body rotation.

$$\boldsymbol{F_c} = \boldsymbol{C}(\boldsymbol{v})\boldsymbol{v}, \qquad \boldsymbol{C}(\boldsymbol{v}) = \begin{bmatrix} 0_{3\times3} & S(\boldsymbol{M}_{11}\boldsymbol{v} + \boldsymbol{M}_{12}\boldsymbol{\omega}) \\ S(\boldsymbol{M}_{11}\boldsymbol{v} + \boldsymbol{M}_{12}\boldsymbol{\omega}) & S(\boldsymbol{M}_{2}1\boldsymbol{v} + \boldsymbol{M}_{22}\boldsymbol{\omega}) \end{bmatrix} \tag{2.3}$$

This first expression in Equation represents the Coriolis force, where $\boldsymbol{C}(\boldsymbol{v})$ is known as the *Coriolis matrix*. This matrix can be directly obtained from the inertia matrix [134]. In the previous expression, $\boldsymbol{M_{ij}}$ $(i, j = 1, 2)$ represent the 3×3 submatrices of the total inertia matrix, M, and S corresponds to the skew-symmetric matrix operator.

Gravity and buoyancy forces
The principle of Archimedes describes the *buoyancy* or aerostatic lift force, which corresponds to the total weight of air displaced by the platform. Since the gravity force and the buoyancy point in opposite directions –downward and upward, respectively– they keep the blimp upright. The amplitude of these forces is expressed as:

$$F_g = mg, \quad F_b = \rho V g, \quad withV = \frac{2}{3}\pi a b^2, \tag{2.4}$$

where m is the mass of the blimp, g the gravitational acceleration of the Earth, ρ the air density and V the volume of the platform. The vector containing the buoyancy and gravitational induced forces and moments corresponds to:

$$\boldsymbol{F_g} = \begin{bmatrix} -(F_g - F_b)s\theta \\ (F_g - F_b)c\theta s\phi \\ -(F_g - F_b)c\theta c\phi \\ -b_z F_b c\theta s\phi \\ -b_z F_b s\theta \\ 0 \end{bmatrix}, \tag{2.5}$$

with b_z denotes the distance between C_g and C_b.

Aerodynamic damping
Aerodynamic damping, also referred to as air friction, mainly depends on the speed of the blimp. The first expression in Equation 2.6 corresponds to the aerodynamic damping vector, where $\boldsymbol{D}(\boldsymbol{v})$ is the damping matrix. According to [178], this matrix contains linear and quadratic damping coefficients to account for both the friction due to the laminar boundary layers (linear) and that caused by the turbulent boundary layers (quadratic).

$$\boldsymbol{F_a} = \boldsymbol{D}(\boldsymbol{v})\boldsymbol{v} \qquad \boldsymbol{D}(\boldsymbol{v}) = -diag\begin{bmatrix} D_{v_x} + D_{v_x^2}\,|v_x| \\ D_{v_y} + D_{v_y^2}\,|v_y| \\ D_{v_z} + D_{v_z^2}\,|v_z| \\ D_{\omega_x} + D_{\omega_x^2}\,|\omega_x| \\ D_{\omega_y} + D_{\omega_y^2}\,|\omega_y| \\ D_{\omega_z} + D_{\omega_z^2}\,|\omega_z| \end{bmatrix} \tag{2.6}$$

Even though this model is an approximation, it is sufficiently accurate for the present case; this is, a highly symmetrical ellipsoid hull and a low operation velocity [57].

Propulsion
Equation 2.7 illustrates the propulsion vector $\boldsymbol{F_p}$ corresponding to a blimp with four thrusters, as shown in Figure 2.4. This mathematical expression can easily be adapted to a blimp with a fewer number of thrusters, as long as they are static, just by eliminating the corresponding forces from the equation.

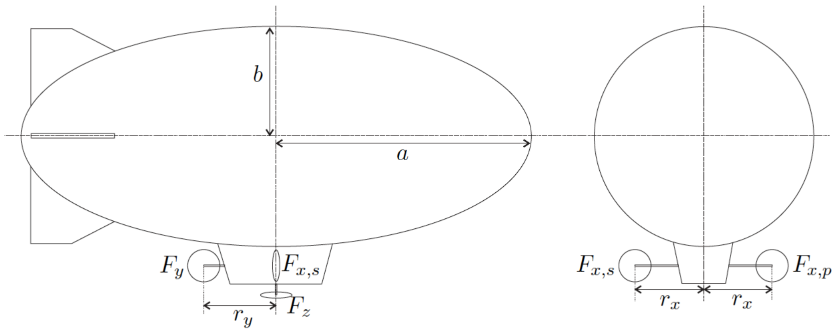


Figure 2.4: Diagram showing the location of the blimp's thrusters. Image from [39].

$$\boldsymbol{F_p} = \begin{bmatrix} F_{x,s} + F_{x,p} \\ F_y \\ F_z \\ 0 \\ 0 \\ F_y r_y + (F_{x,s} - F_{x,p}) r_x \end{bmatrix}, \tag{2.7}$$

where r_y is the distance between C_g and thruster F_y along the x-axis and r_x the distance between the lateral thruster $F_{x,s}$, $F_{x,p}$ and C_g along the y-axis.

2.2.1. Full dynamic model equations

Finally, the full mathematical model of the blimp based on the previously analyzed models is given by:

$$\begin{aligned}
\dot{x} &= c\psi c\theta v_x + (s\psi c\phi + c\psi s\theta s\phi) v_y + (s\psi s\phi - c\psi s\theta c\phi) v_z \\
\dot{y} &= -s\psi c\theta v_x + (c\psi c\phi - s\psi s\theta s\phi) v_y + (c\psi s\phi + s\psi s\theta c\phi) v_z \\
\dot{z} &= s\theta v_x - c\theta s\phi v_y + c\theta c\phi v_z \\
\dot{\phi} &= \omega_x + s\phi t\theta \omega_y + c\phi t\theta \omega_z \\
\dot{\theta} &= c\phi \omega_y - s\phi \omega_z \\
\dot{\psi} &= s\phi / c\theta \omega_y + c\phi / c\theta \omega_z \\
\dot{v}_x &= \frac{1}{m'_x}(-m'_z v_z \omega_y + m'_y v_y \omega_z - (F_g - F_b) s\theta - (D_{v_x} + D_{v_x^2} |v_x|) v_x + (F_{x,s} + F_{s,p})) \\
\dot{v}_y &= \frac{1}{m'_y}(m'_z v_z \omega_x - m'_x v_x \omega_z + (F_g - F_b) c\theta s\phi - (D_{v_y} + D_{v_y^2} |v_y|) v_y + F_y) \\
\dot{v}_z &= \frac{1}{m'_z}(-m'_y v_y \omega_x + m'_x v_x \omega_y + (F_g - F_b) c\theta c\phi - (D_{v_z} + D_{v_z^2} |v_z|) v_z + F_z) \\
\dot{\omega}_x &= \frac{1}{I'_x}((m'_y - m'_z) v_y v_z + (I'_y - I'_z) \omega_y \omega_z - b_z F_b c\theta s\phi - (D_{\omega_x} + D_{\omega_x^2} |\omega_x|) \omega_x) \\
\dot{\omega}_y &= \frac{1}{I'_y}(-(m'_x - m'_z) v_x v_z - (I'_x - I'_z) \omega_x \omega_z - b_z F_b s\theta - (D_{\omega_y} + D_{\omega_y^2} |\omega_y|) \omega_y) \\
\dot{\omega}_z &= \frac{1}{I'_z}((m'_x - m'_y) v_x v_y + (I'_x - I'_y) \omega_x \omega_y - (D_{\omega_z} + D_{\omega_z^2} |\omega_z|) \omega_z + (F_y r_y + (F_{x,s} - F_{x,p}) r_x))
\end{aligned} \tag{2.8}$$

where $s\phi = \sin(\phi)$ and $c\phi = \cos(\phi)$.

2.3. Simulation on the ROS/Gazebo framework

Since robotic airships are just starting to gain popularity again, not many open-source Gazebo blimp simulators are available online. Therefore, the most common approach that is taken when doing any kind of project that involves a blimp robot is to build the actual platform and perform the tests in a real-world environment. There are, however, two main works that should be highlighted regarding simulator environments for blimps.

First, in [151], a methodology to simulate both the control and dynamics of an airship by means of the virtual platforms ROS and Gazebo is thoroughly delineated. Concretely, the motion control algorithm is developed in ROS and the dynamics of the blimp are implemented in Gazebo. Moreover, velocity hold, altitude hold and pitch regulation of the airship are simulated using PID control. Nevertheless, even if the whole process to generate the simulation is described in a very detailed way, this project is not open-source, so that all the coding to replicate similar results should be carried out from scratch.

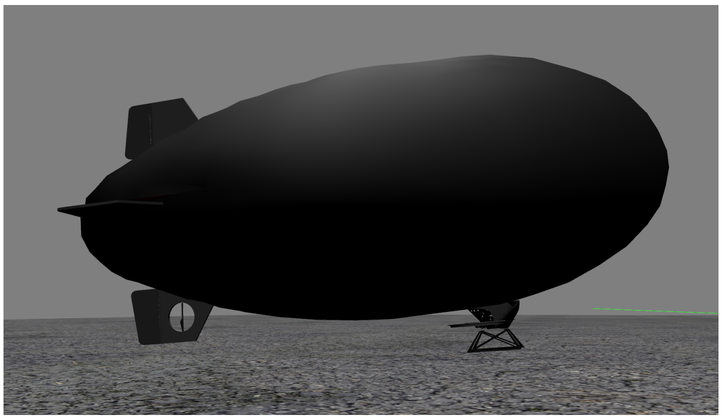


Figure 2.5: Blimp robot simulated in Gazebo[2]. Image from [2].

On the other hand, a completely open-source and very interesting radio-controlled (RC) blimp simulator developed at the *Max Planck Institute for Intelligent Systems* is presented in [2]. All the necessary files, packages and plugins to properly simulate the dynamics of the blimp in Gazebo are provided and accompanied by detailed instructions on how to use each of the files, configure the dependencies and perform the required installations. Figure 2.5 illustrates the design of the RC blimp modelled within this framework.

[2]https://github.com/ootang2018/blimp_simulator

3
Remote sensing in UAVs: radars

The most widespread remote sensing strategy that has been used in recent years to capture UAVs close vicinity mainly involves vision-based sensors. Nevertheless, a notable increase of radar-based navigation has been lately taking place, due to the great potential that radars have to offer when it comes to overcoming the limitations posed by monocular and stereo vision cameras. This chapter provides an overview of the existing and most widely used remote sensing methodologies in the realm of UAVs navigation in Section 3.1, also analyzing the pros and cons of each alternative. In Section 3.2, the fundamentals of FMCW radar-based sensing are presented, which correspond to the sensory strategy employed in this work. Therefore, its working principle, together with the mathematical background and related signal processing algorithms are explained.

3.1. Remote sensing strategies in UAVs

The good maneuverability, hovering and stable flight properties that are inherent to contemporary consumer and industrial unmanned aerial vehicles make them great easy-to-use flying sensor platforms [83]. In this way, two main objectives can be differentiated when it comes to remote sensing in UAVs: stabilization and localization, or direct environment sensing. In the former case, sensors such as inertial measurement units (IMUs), barometric sensors and the Global Navigation Satellite System (GNSS) are utilized. However, when it comes to fully automated flight, object detection or collision avoidance, this is, the latter case, additional sensory strategies must be used.

3.1.1. Vision-based sensors applications

Very common applications of vision-based sensors such as visible-light and thermal cameras are encountered within industrial environments, where UAVs are employed to perform structures, wind turbines or power lines inspections [90]. These kinds of sensors are also widely used in forestry, to estimate species diversity [176] or tree canopy heights [98]. Applications can also be found within agricultural industry, which corresponds to the largest commercial market for UAVs at a global scale [124]. For example, UAVs can take images of vasts amounts of terrain at a higher resolution than satellites, even under cloudy conditions [83], and are able to monitor plant health and field conditions. Despite of their multiple properties and wide usage, vision-based sensors also have their limitations. For example, their performance strongly relies on the lightning conditions, the reflectivity of the material and they struggle with regular surface patterns.

3.1.2. Radar sensors advantages and applications

Radars, on the other hand, are not influenced by variations in illumination and are able to detect obstacles for collision avoidance even with backlight present and under severe low-contrast circumstances [83]. Their ability to directly sense both speed and range is also one of their most advantageous features. Angular information also be extracted with the help of a multichannel radar, which is the case of this project. Furthermore, the rising use of radar in the automotive industry and its promising results [70][164] has led to a noteworthy decrease in its costs. Thanks to the production of single package radar front ends, also including antennas, in the medical, industrial and scientific bands for industrial applications [68][11][76], the wonder of lightweight radar sensors valid for UAV applications has come to be true.

These interesting features have significantly produced a remarkable increase in the usage of radar-based sensors within UAVs navigation in recent years. For example, in [82] a detailed analysis of a 77-GHz radar-based altimeter is presented. In [143], a platform using a 24-GHz frequency-modulated continuous wave (FMCW) radar for full interferometric SAR imaging and processing is evaluated. Since both velocity and distance can be measured with a FMCW radar, a 24-GHz FMCW radar with 1-GHz bandwidth for potential intruder and ground target detection was also studied in [137]. In this case, however, the device was not mounted on the actual UAV but on a ground station. An additional 24-GHz FMCW radar and its blend with accelerometer data were tested indoors by means of a motion-capture system made of 24 cameras, in [104].

Table 3.1 shows a summary of the strengths and weaknesses of each of the type of sensors analyzed so far in this chapter.

Sensor	Accuracy	Weather Dependancy	Light Sensitivity	Range	Processing Required	Power Needed
Radar μ	High	None	No	Long	Low	High
Radar mm	High	Low	No	Long	Low	Medium
Radar K	High	Medium	No	Medium	Low	Low
Ultrasonic	Medium	Medium	No	Short	Low	Medium
Thermal/IR	Medium	High	No	Medium	High	Low
Camera	Medium	High	Yes	Short	High	Low

Table 3.1: Sensor attribute comparison. Table modified from [175].

3.2. FMCW radar and its signal processing

The high integration potential inherent to FMCW radars have made them shine in recent years. Even though the automotive industry is the main driver in developing these small imprint solutions, a gradual consolidation into other arenas is becoming evident. In this section, the working principle of FMCW radars and some of the most relevant algorithms to measure distance, velocity and azimuth are presented.

3.2.1. Working principle

The basics of radars

Before diving into discussing the details of a FMCW radar, the basic working principal of any radar configuration in general must be understood. In his book *Introduction to Radar Systems* [147], Skolnik writes:

> *Radar is an electromagnetic system for the detection and location of reflecting objects such as aircraft, ships, spacecraft, vehicles, people and the natural environment. It operates by radiating energy into space and detecting the echo signal reflected from an object or target. The reflected energy that is returned to the radar not only indicates the presence of a target, but by comparing the received echo signal with the signal that was transmitted, its location can be determined along with other target-related information.*

Concretely, two main types of remote sensing radars can be distinguished:

1. The pulsed radar, which transmits short signals, and then *listens* for the echoes during the time intervals between the transmit pulses. The pulse repetition frequency (PRF) f_p and length t are selected so that the desired return signal can be received without interference from the signal that is being transmitted.

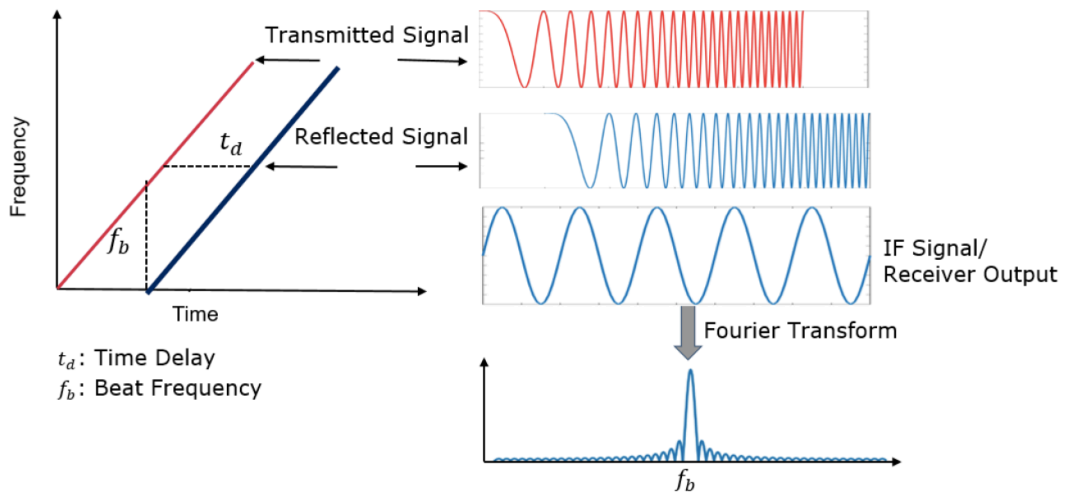


Figure 3.1: Basic principle of operation of the FMCW radar system. Image from [4].

2. The continuous wave (CW) radar, which corresponds to the simplest existing type of radar. As its name already suggests, it transmits a continuous wave signal which reflects or *scatters* off the target to the receive antenna. A CW radar, however, has no range- or timing-discrimination capability [160]. This occurs because its time resolution is equivalent to an infinitely long pulse length. By applying a proper frequency modulation (FM) strategy to the transmit signal, this issue can get solved.

The fundamental signal of the FMCW radar

In FMCW radars the signal is continuously transmitted and received, but its frequency is modified as a function of time, as it can be seen in Figure 3.1. The term that is most commonly used to denote these kind of signals is *chirp* and sometimes, although less often, *sweep*. Therefore, a chirp refers to any sinusoid whose frequency increases (up-chirp) or decreases (down-chirp) over time.

Concretely, Figure 3.1 exemplifies a linear chirp, since the instantaneous frequency f has a linear dependency with time t, as follows:

$$f(t) = f_0 + \frac{B}{T_c}(t - t_0) = f_0 + S(t - t_0) \ , \tag{3.1}$$

where f_0 corresponds to the starting frequency at time $t = t_0$, B is the chirp bandwidth, T_c the chirp time, also referred to as the modulation time and S denotes the rate of frequency change (frequency slope) or chirpyness. The time-domain function for the phase, ϕ, can simply be obtained by integrating the angular frequency, $\omega(t) = 2\pi f(t)$:

$$\phi(t) = \phi_0 + 2\pi \int_{t_0}^{t} f(\tau) d\tau = \phi_0 + 2\pi \left[f_0(t - t_0) + \frac{B}{2T_c}(t^2 - t_0^2) \right] \tag{3.2}$$

The corresponding time domain function for a linear chirp is, according to [116], the sine of the quadratic-phase signal in radians:

$$y_c(t) = v_{TX}(t) = A_c \sin\left(\phi_0 + 2\pi f_0 t + \pi \frac{B}{T_c}(t - mT_c)^2 \right) \ , \tag{3.3}$$

where A_c represents the chirp's amplitude, m corresponds to the m^{th} sweep and $t_0 = 0$ by taking the assumption that chirps take place continuously.

Finally, the modulation bandwidth and starting frequency, B and f_0, give rise to one more important parameter: the carrier frequency, f_c. This parameter which is defined as $f_c = f_0 + B/2$, represents the central frequency for the spectrum band that is being covered. In this work it corresponds to 24 GHz. Common bandwidth spans are normally in the order of up to several GHz, while sweep modulation times typically belong to the range of dozens of microseconds up to a millisecond.

The FMCW radar operating principle

The procedure from which these type of signals (see Figure 3.1) are generated is illustrated in Figure 3.2, which represents a simplified block diagram of a FMCW radar. First, a FMCW synthesizer generates an appropriate chirp signal. Then, the power amplifier (PA) amplifies the generated sweep, which is later transmitted by the transmit antenna. The receive antenna captures chirps that are reflected back from objects or obstacles and the received signal is passed through the low-noise amplifier (LNA). Finally, a down-conversion frequency mixer conflates the RX and TX signals and their inputs to generate an intermediate frequency (IF) signal at its output. The shape of all the RX, TX and IF signals can be observed in Figure 3.1. The latter is also referred to as the beat frequency and contains useful information about the irradiated targets. The beat signal is then low-passed filtered and sampled by an analog to digital converter (ADC).

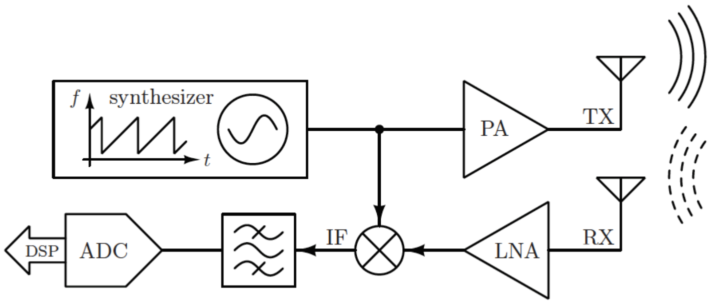


Figure 3.2: Block diagram of a typical homodyne FMCW radar by [116].

It is noteworthy to highlight that the RX signal is just a time-delayed replica of TX. The IF signal will have a constant frequency proportional to the reflected signal round-trip delay, which can be identified by spectral analysis (e.g. FFT):

$$f_{IF} = f_b = \frac{2r}{c}S = \frac{2B}{cT_c}r \ , \tag{3.4}$$

where r corresponds to the distance between the radar and the object, c is the speed of light and S represents the proportionality coefficient.

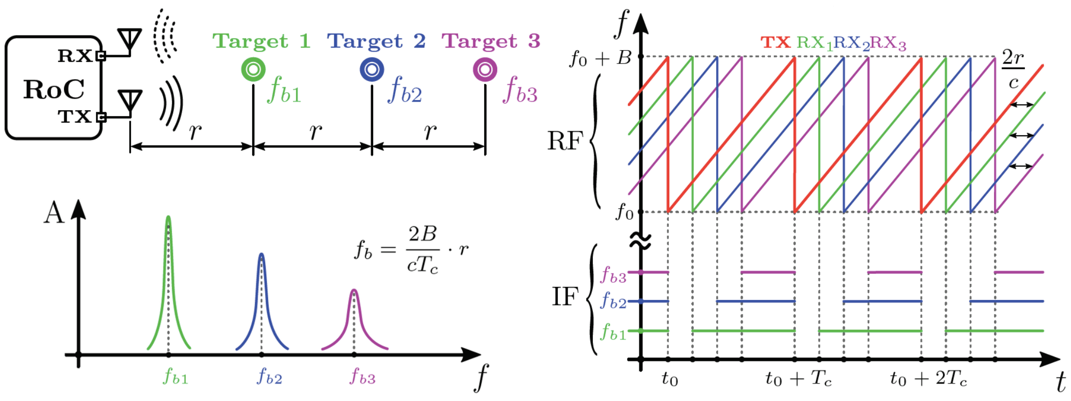


Figure 3.3: Static multitarget detection with an FMCW radar by [116].

When multiple targets located at different distances are part of a scene, a single transmitted chirp will generate multiple received chirps, each with a different time delay depending on the distance to each particular obstacle. Thus, as explained by [116], the produced IF signal will be composed of several frequency tones, each of which is directly proportional to the range of each of the objects. This phenomenon is illustrated in Figure 3.3.

3.2.2. Extracting range and velocity: the Range-Doppler algorithm

Range and velocity estimation can be easily computed by means of a two dimensional fast Fourier transform (2D FFT) in what is known as the Range-Doppler algorithm, which is graphically represented in Figure 3.4. In this section, the underlying details of this algorithm will be discussed. Concretely, special focus will be put on the specific set of steps that the sensor *Infineon Position2Go*[1], which corresponds to the one employed in this work, takes in order to obtain the range and velocity of the detected targets.

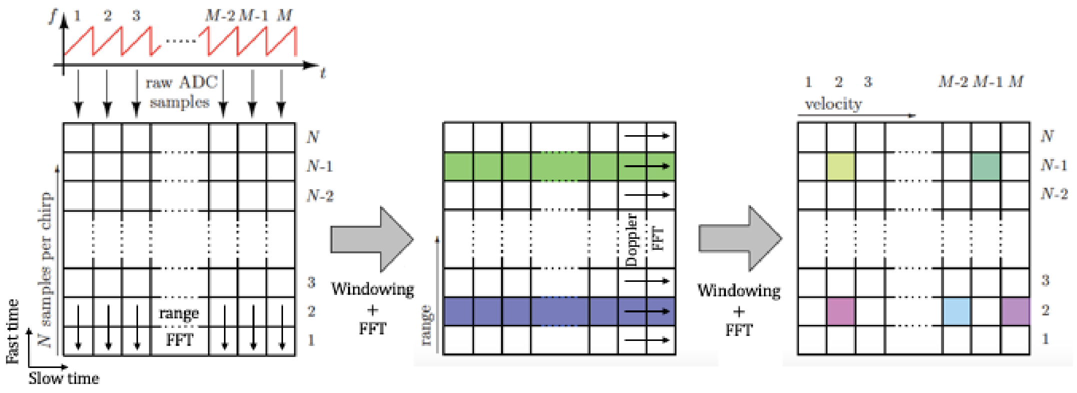


Figure 3.4: 2D FFT processing of an FMCW frame containing M chirps and that N samples are taken out of each chirp. Modified from [116].

First of all, a multi-chirp signal is transmitted for every frame (see Figure 3.5), reflected by the objects present in the surroundings and received, following the procedure previously explained (see Figure 3.2). For each of these chirps present in the multi-chirp transmitted signal, an IF signal is generated and digitally sampled by an ADC, giving rise to a two-dimensional raw data matrix, corresponding to the first step shown in Figure 3.4. The dimensions of this matrix are $M \times N$, with M, the denoting the number of sweeps and N the number of samples taken out of each of the chirps.

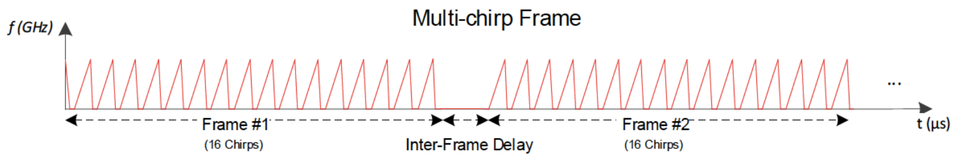


Figure 3.5: Position2Go generated FMCW multi-chirp signal.

Range processing

After the pre-processed raw data for every receiver is available, the processing over the fast time samples is carried out to extract information about the range of the targets. First, a Blackman window is applied across the range dimension, in order to enhance the signal SNR and supress the side lobes. Then, zero padding is used to improve the received signal characteristic. Finally, a FFT is carried out over the range dimension, resulting in an FFT image, corresponding to the second step shown in Figure 3.4. In order to detect target ranges, a peak search is carried out over this range FFT [4]. Moreover, the range is obtained by isolating r from Equation 3.4, as follows:

$$r = \frac{cf_b}{2S} = \frac{cT_c}{2B} \cdot f_b \tag{3.5}$$

Doppler processing

In the case of nonstationary objects and/or radar, all range measurements through round-trip delay are going to be affected by what is commonly known as the Doppler effect: this is, signal elongation or compression, depending on whether the object is moving towards or away from the radar. Concretely, small displacements of an object, Δd, have a clear effect on the IF's phase:

$$\Delta\phi = 2\pi f_0 \Delta t = 2\pi f_0 \frac{2\Delta d}{c} = \frac{4\pi}{\lambda_0} \cdot \Delta d \xLeftrightarrow{\Delta d = vT_c} v = \frac{\lambda_0}{4\pi T_c} \cdot \Delta\phi \ , \tag{3.6}$$

where $\lambda_0 = c/f_0$ accounts for the wavelength of the transmitted RF signal and Δt represents the round-trip delay change caused by the object's range displacement.

[1]https://www.infineon.com/cms/media/PMM_3dmodels/position2go.html

As explained in [116], the key for measuring the speed of a target with a FMCW radar relies on transmitting two consecutive sweeps of duration T_c. A FFT is applied then to each of these two sweeps to obtain the distance to the object. This range-FFT shows a peak at the same position but with a different value of phase. Therefore, the phase difference between two peaks, this is, measured across two consecutive chirps, can be used to estimate the speed of an obstacle surrounding the radar.

This method, however, would not work for the case of multiple moving objects, with different speeds, that are all equidistantly disposed from the radar. The details of why this happens can be found in [116]. To solve this, a series of more than two consecutive equally spaced chirps should be transmitted; this is, a multi-chirp signal, as shown in Figure 3.5. Computing a second FFT, e.g. the Doppler-FFT, across the discrete sequence of equispaced chirps (or frame) would yield peaks corresponding to the normalized angular frequencies, ω, of each target speed. By substituting $\omega = \Delta\phi$, Equation 3.6 can be utilized to calculate the object velocities. It is worth to underscore that the Doppler-FFT can only be performed after all the range-FFT data points have been calculated. The last step in Figure 3.4 presents the two-dimensional grid obtained after computing the second FFT or Doppler-FFT.

Summary of the range-Doppler algorithm
The procedure for calculating the range and velocity of targets through the range-Doppler algorithm can be summarized as follows [72]:

1. First, the received signal at frame n_k, $s_{IF}(t;n_k)$, from consecutive chirps are arranged in the form of a 2D raw dara matrix: $s_{IF}(n_s,n_f;n_k)$, where n_s and n_f denote the slow and fast time index, respectively.

2. The range-Doppler image (RDI) is generated for each channel by applying a window function, zero padding and then a 1D FFT along fast time to obtain the range transformation.

3. Followed by applying another window function, zero-padding and a 1D FFT but, this time, along the slow time index to also obtain the velocity transformation.

4. In the end, the two 1D FFTs, transform the signal $s_{IF}(n_s,nf;n_k)$ along fast and slow time, into the range-Doppler domain:

$$S(p,q,n_k) = \sum_{n_s=1}^{Z_{N_c}} \left(\sum_{n_f=1}^{Z_{NTS}} w_f(n_f) s_{IF}(n_s,n_f;n_k) e^{-j2\pi p n_f / Z_{NTS}} \right) \cdot w_s(n_s) e^{-j2\pi q n_s / Z_{N_c}} , \quad (3.7)$$

   with NTS and Z_{NTS} being the number of transmitted samples defined by the DAC sampling points over the chirp duration and zero padding along fast-time, respectively. N_c and Z_{N_c} being the number of chirps in a frame and zero padding along the slow-time, respectively. $w_f(n_f)$ and $w_s(n_s)$ denote the window function along fast and slow-time, respectively. In the case of the *Infineon Position2Go* radar, which is the one related to this work, $w_f(n_f)$ corresponds to a Blackman window and $w_s(n_s)$ to a Chebyshev window. Finally, p and q represent the index over range and Doppler, respectively.

5. The peaks in the RD domain, using saw-tooth FMCW with fast ramps (see Figure 3.5), occur at:

$$p_k = \left(\frac{2B}{cT} r_k \right), \quad q_k = \frac{2v_k f_c}{c} \quad (3.8)$$

Additional important parameters
Four relevant parameters that have to be taken into account when measuring with a FMCW radar are range and velocity resolution and maximum measurable speeds and distances. Mathematically, two or more nonidentical IF signal tones can be resolved as long as $\Delta f > 1/T_c$. With this and Equation 3.4 in mind, the expression for the radar's range resolution yields [160]:

$$\Delta r > \frac{c}{2ST_c} = \frac{c}{2B} \quad (3.9)$$

Moreover, the maximum range that a FMCW radar can detect is corresponds to:

$$r_{max} = \frac{cf_s}{4S} = \frac{cT_c f_s}{4B} = \frac{cN}{4B} \ , \tag{3.10}$$

where f_s is the maximum ADC's sampling rate according to Figure 3.2 and N the number of ADC samples per chirp. In the case of the velocity, two different normalized angular frequencies can be differentiated provided that $\Delta\omega > 2\pi/M$. Accounting for this and Equation 3.6 the speed resolution can be found:

$$\Delta v > \frac{\lambda_0}{2MT_c} = \frac{\lambda_0}{2T_f} \ , \tag{3.11}$$

where M is the number of observed samples, T_c the separation between consecutive sweeps and $T_f = MT_c$, the fram duration. Besides, the phase measurement is unambiguous only in cases in which $|\Delta\phi < \pi|$ and the maximum unambiguous measurable speed corresponds to:

$$v_{max} = \frac{\lambda_0}{4T_c} \tag{3.12}$$

3.2.3. Angle measurements

Resolving the angular dimension of objects can be achieved by measuring the IF signal's phase change over multiple antennas separated in space. Concretely, in the case of the *Infineon Position2Go* radar, which comprises 1 TX and 2 RX antennas and is the one used in this work, a phase monopulse angle estimation method is used to estimate the Angle of Arrival (AoA) of targets from the range-Doppler map. Therefore, instead of calculating the phase difference between consecutive frames (see Equation 3.6), as it is done to obtain the velocity, the phase difference over the antennas RX_1 and RX_2, separated in space, is computed to finally obtain the target angle. The required process to calculate the AoA is illustrated in Figure 3.6.

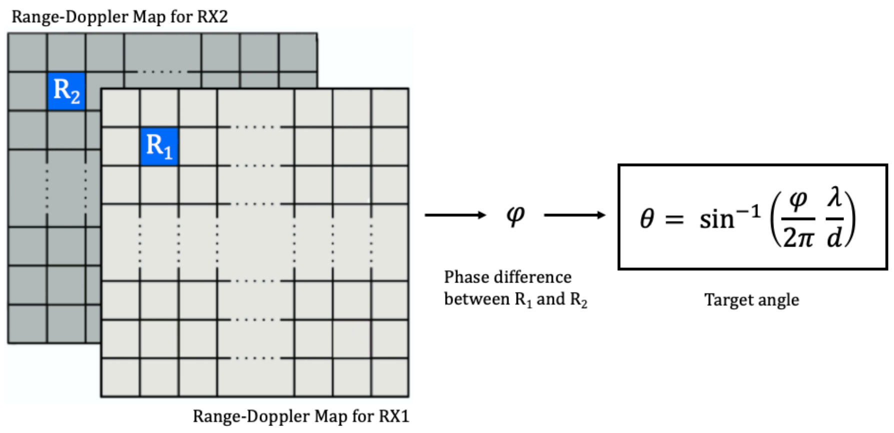

Range-Doppler Map for RX2
R_2
R_1
φ
$\theta = \sin^{-1}\left(\frac{\varphi}{2\pi}\frac{\lambda}{d}\right)$
Phase difference between R_1 and R_2
Target angle
Range-Doppler Map for RX1

Figure 3.6: Monopulse angle estimation method flow.

Therefore, the first step is to compute the phase difference between R_1 and R_2, which correspond to the value of the range-Doppler map at the target range highlighted in blue for RX1 and RX2, respectively. Then, with this phase difference, ϕ, the target angle can be calculated by means of the following expression:

$$\theta = \sin^{-1}\left(\frac{\phi}{2\pi} \cdot \frac{\lambda}{d}\right) \ , \tag{3.13}$$

where λ denotes the wavelength of the incident signal and d corresponds to the antenna spacing: in this case, the spacing between RX1 and RX2.

4 Blimp control strategies

By using blimp robots as experiment platforms to study robot control, navigation algorithms and information acquisition, researchers have put increasing attention on developing control-based applications for autonomous indoor airships over the last few years. Most of these applications, however, are based on classical control approaches such as PID controllers, which offer less flexibility than their knowledge-based counterparts when it comes to vision-based navigation, signal processing, fault tolerant or adaptive control strategies among many others. In this section, some of the most common classical control strategies applied to blimps are briefly analyzed to then put special focus on some interesting and innovative knowledge-based control works.

Methodology	**Task**	**Reference**
Classical approaches		
PID	Trajectory control	[165]
PID	Altitude and heading control	[66]
PID	Altitude and heading control	[33]
PID	Altitude and heading control	[162]
Predictor-based	Altitude control	[170]
Predictor-based	Altitude control	[169]
Knowledge-based approaches		
GP and RL	Yaw control	[97]
RL	Altitude control	[132]
Fuzzy logic and PID	Altitude control	[63]
Artificial evolution	Trajectory control	[178]
Neuromorphic controller	Altitude, drift, obstacle avoidance	[14], [126]

Table 4.1: Brief summary of different indoor blimp control strategies found in literature.

4.1. Classical control

First of all, the design, modeling, linearization and control law development of a solar-powered blimp is proposed in [165]. Concretely, a PID attitude and position controller is developed in simulation and tested in an indoor environment verifying that the blimp is able to gather energy and maintain its stability while controlling its flight. In [66], as already analyzed in Section 2, an open-source, low-cost, robotic airship with a 3D-printable gondola for education and research is developed and tested for proof-of-concept purposes in a basic altitude PID control and path following task, proving its efficiency. On a similar note, the GT-MAB, one of the smallest indoor robotic airships that have been developed [33], is tested on both a way-point and altitude control task. In the former case, a common PID controller is implemented and, to maintain the desired height, a PID controller with two different sets of gains combined with a scheduling algorithm that decides which of them to use is developed. In the work by Van der Zwaan [162], a PID control law is proposed for the lateral and longitudinal motion on a vision based station keeping and docking for an aerial blimp. In the works by Wang [170][169], the dynamics of the utilized robotic blimp are derived, together with its parameter identification and an accurate altitude controller based on a high-order sliding mode (HOSM) differentiator as an observer to assess the vertical velocity and a predictor-based controller for height stabilization. The performance of the controller is verified both on a simulated environment by means of Simulink and also on the real airship with the help of an OptiTrack capturing system.

4.2. Knowledge-based control

Since most of the developed control approaches employed with blimps belong to the realm of classical control, the works presented in the previous subsection are just some interesting related examples from the many that exist. In the case of knowledge-based strategies concerning robotic airships, however, not many works have been published. In this section, some of the most relevant ones, to the best of the author's knowledge, are reviewed, following Table 4.1.

4.2.1. Reinforcement learning applied to yaw and altitude control

As opposed to classical system modeling techniques in which the system is represented as an ordinary differential equation (ODE) based on Newtonian laws, a GP-enhanced identification model that provides an estimate of uncertainty in addition to offering improved state predictions than those of either ODE or GP alone is proposed in [97]. Moreover, reinforcement learning and optimal control are used in combination with this GP-enhanced model in order to steer the blimp from any yaw ψ and yaw rate $\dot{\psi}$ to a goal yaw ψ^* with zero yaw rate. Along the same lines, a model-free reinforcement learning strategy is applied to the problem of learning height control policies for aerial blimps in [132]. Concretely, in this work the blimp is able to learn the policy online –directly on the platform–, within a few minutes, without a predefined physical model of the dynamics, and is able to deal with the continuous state-action space provided by the current estimate for the altitude and the vertical speed.

4.2.2. Fuzzy logic for altitude control and navigation tasks

In [63], a fuzzy controller with two inputs: the height error and the estimated current vertical velocity, is proposed to control the altitude of the blimp. This approach is characterized by a set of linguistic variables and fuzzy if-then rules which provide the necessary knowledge to the fuzzy controller. Following the same strategy, a collision avoidance controller is also implemented and both of them are compared with the classical PID approach.

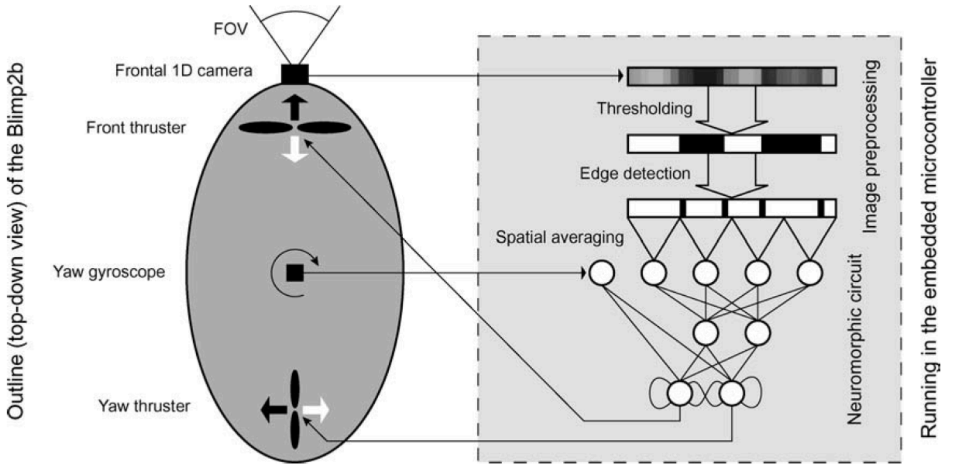


Figure 4.1: Neuromorphic approach by [178]. Blimp sensors and actuators (left) and NN architecture (right).

4.2.3. Neuromorphic strategies for trajectory and height control

In [178], a simple neuromorphic controller, based on [120], is first evolved in simulation to map visual inputs to motor commands to steer the blimp and avoid obstacles in a vision-based navigation task. Then, the best individuals are transferred to the physical blimp so that the approach can also be tested in a real-world environment. Regarding the architecture of the NN, which is illustrated in Figure 4.1, it has four input neurons that receive visual information, one input unit connected to the rate gyro, one bias unit, two hidden neurons and two output neurons with recurrent and lateral connections driving the yaw and frontal thrusters. Finally, in [14], a simple neuronal control system for a blimp-based UAV that offers course stabilization, drift and altitude control and collision avoidance trying to imitate the effectiveness of insects' visual system is proposed.

5 Neuromorphic computing for radar-based navigation

Once the proper mathematical processing required to obtain the Range-Doppler Image (RDI), as explained in Section 3, is clear, some kind of control strategy should be used to develop an accurate altitude controller based on the available data, as proposed in Section 4. In this work, a neuromorphic approach based on spiking neural networks (SNN) is used for this purpose, given the favorable properties that this strategy offers in terms of low power consumption, online learning, fast inference, massive parallelism and event-driven processing. However, the many breakthroughs that robotics control based on learning-inspired SNN has accomplished in recent years, has only been possible because they stand on the shoulders of its giant predecessors, ANN, which are still the main protagonists in the field of robotics control. Therefore, diving into the world of SNN inherently means addressing the previous generation of neural networks, ANN.

Following a chronological order, this chapter starts by introducing the underlying theory behind artificial neural networks, together with some of the most common architectures and their application to the field of robotics control. Afterwards, the equivalent areas are analyzed for the case of the newer spiking neural networks.

5.1. Artificial Neural Networks

Artificial neural networks (ANNs) constitute computing systems that are vaguely inspired by the biological neural networks present in animal brains. Concretely, each ANN is built upon a collection of connected units referred to as *neurons*, which very loosely model the neurons in a biological brain. Each of these connections can transmit a signal to other nodes, mimicking the synapses that take place in realistic brains. Since *error backpropagation* was proposed as an efficient learning algorithm [133], this computational framework has been used in an infinite range of applications ranging from medicine or economics to engineering. Due to the extensive literature available on artificial neural networks, this section puts special attention on the relevant aspects, common architectures and strategies for flight control and deep learning applications for UAVs. For more general information on artificial neural networks, please refer to [42], [64].

5.1.1. Theoretical background and basic components

The computation performed by artificial neurons can be divided in two main steps: the weighted sum of their inputs and a non-linear operation from which a neuron generates an output in case that the inputs surpass a certain threshold. Based on Figure 5.1, the activation of the output neuron is given by:

$$y_j = f\left(\sum_{i=1}^{p} w_{i,j} \cdot x_i\right), \tag{5.1}$$

where p represents the number of presynaptic neurons, $w_{i,j}$ the synaptic efficacies and x_i the activation values. The synapses can be categorized as excitatory or inhibitory depending on the sign of $w_{i,j}$. In order to introduce non-linearities into the network, the *activation function* $f(\cdot)$ is applied after the weighted sum. The sigmoid, hyperbolic tangent or Rectified Linear Unit (ReLU) [119] are some examples of commonly used activation functions. Figure 5.1 illustrates the structure of one of the most common artificial neural network architectures: the Multilayer Perceptron (MLP). In this scheme, a very simple network with only three layers is presented, each of them represented with a different color. First, in red, the input values that must be processed for the generation of the desired output are passed onto the *input layer*. Next, these values are propagated to the next stage of neurons, known as the *hidden layer*. Finally, the outermost layer or *output layer* performs the last processing steps and eventually presents the results to the user.

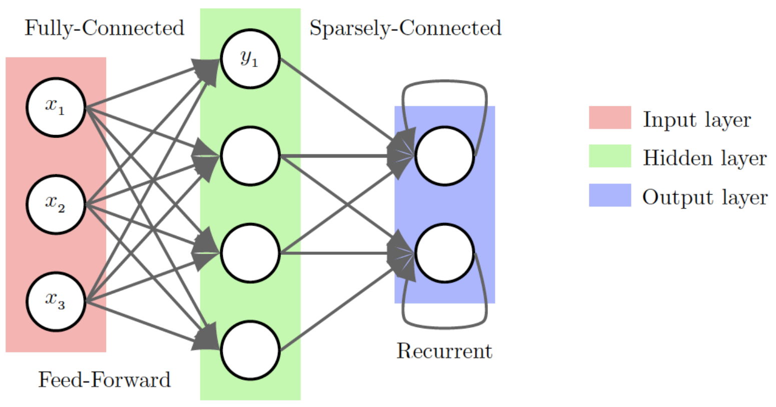


Figure 5.1: Diagram of a simple ANN. Image from [122].

Apart from the the number of neurons and layers, the different types of connections that are defined between the neurons in the network also have a huge impact on the type of computation performed by the model and its complexity. Figure 5.1 also shows the three different existing neural connections. From left to right, it can be seen that in a *fully-connected* layer all presynaptic neurons have synapses with all the postsynaptic cells. This is not the case for the *sparsely-connected* version, where only some synapses are present, but not all pre and postsynaptic neurons are directly connected. These two types of connections also belong to a greater category known as *feed-forward* connections, where the outputs are always derived from a sequence of operations on the activations of neurons from previous layers. Then, in the case of the *recurrent* layer, the network is able to exhibit temporal behavior, since the output activation from a certain neuron is reused as an input to that same cell exactly one timestep later. Therefore, as a final note, recurrent networks normally work with sequential input data, while the feed-forward version assumes that this information is uncorrelated.

For the introduction to the basics of artificial networks, this work has referred to the theoretical knowledge collected by [105], [152] and [122].

5.1.2. Deep learning architectures

Neural architectures as the one illustrated in Figure 5.1, this is, with very few hidden layers, are commonly denoted as *shallow networks*. Although successfully trained with supervised learning rules like *error backprogagation*, the low complexity of the input-output mapping approximations that can be obtained with these kind of networks is very limited. This is why the study of learning methods that could be applied to neural architectures with higher complexity has been a hot research topic since [133]. This line of work is referred to as *Deep Learning* (DL) and the architectures used within it are known as *Deep Neural Networks* (DNN). Moreover, the recent success of DL techniques in machine learning or pattern recognition has highly been due to the availability of increasing computational capacity, open-source networks and vasts amounts of information [141]. In this section, the fundamentals of the most widely used deep learning technologies currently available in supervised, unsupervised, and reinforcement learning are covered. For more detailed information regarding the different deep learning architectures, please refer to [105].

Supervised learning
Given a training a proper training set with examples, supervised learning (SL) algorithms learn to associate a given input with some output with a certain probability. Some of the most well-known algorithms in SL include feedforward neural networks, convolutional neural networks (CNN), recurrent neural networks (RNN) and long short-term memory models (LSTM). This subsection analyses the basic concepts of each of these architectures:

1. Feedforward neural networks or MLPs: given a sample vector $\boldsymbol{x}$ with n features a trained algorithm is expected to compute an output value $\boldsymbol{y}$ that is consistent with the input-output mapping provided in the training set. The layers of these networks consist of neurons whose activation, given a certain input $x \in \mathbb{R}^n$, is given by $a_\theta(x) = g\left(\theta^T x\right)$, where g represents the chosen activation function and θ a vector of n weights. During the learning process these weights are updated through backpropagation.

2. Convolutional neural networks (CNNs): these models take their name from the mathematical linear operation of convolution, which is always present in one or more layers of the network. They accept two-dimensional input data, like time data or images. Computer vision is one of the most relevant applications of CNNs.

3. Recurrent neural networks (RNNs): as already seen in Figure 5.1, the outputs of these networks are a function of the current inputs but also of the previous outputs. The learning algorithm most commonly used in RNNs is an extension of backpropagation, known as *backpropagation through time* (BPTT), which takes into account temporality when computing the gradients.

4. Long-short term memory (LSTM): these models are a type of RNN architecture which successfully handles the issue of vanishing gradients and allows for continuous learning over a larger number of time steps. Speech recognition is one of the most successful applications of LSTM.

Unsupervised learning
In contrast to the previous section, the purpose of unsupervised learning consists in developing models that are capable of extracting meaningful representations from high-dimensional sensory unlabeled data [159]. The visual cortex, which only requires a very small amount of labeled data, has served as an inspiration to this learning process. Concretely, models such as the deep belief networks (DBNs) [75] [12] allow the unsupervised learning of several layers of nonlinear features. To train these type of models, the learning usually takes place by applying a *contrastive divergence* algorithm [74]; the details of which follow out of the scope of this literature study. DNN can also be used for dimensionality reduction of the input data.

5.1.3. Deep learning applications for UAVs

The vast amount of deep learning applications in the field of robotics and, concretely, within the UAVs arena is unfathomable. Moreover, analyzing them in detail falls beyond the scope of this work, as the focus in this study will mainly be put in the latest generation of NNs, spiking neural networks, due to their more favorable properties in terms of event-driven processing, online learning, low power consumption and massive parallelism. Nevertheless, some very interesting reviews that summarize most of the applications that combine DL and UAVs are provided in this section, for the interest of the reader. First, in [67], a detailed survey of UAV model-based flight control strategies with ANNs is presented. The type of ANN architecture, together with the efficiency, training and control approach or the kind of UAV (fixed-wing, helicopter or quadrotor) used in the different works published in recent years are analyzed. Then, an interesting review of different DL methods and applications for UAVs is proposed in [159]. Concretely, DL techniques for feature extraction, planning and situational awareness, and motion control are studied. Finally, a review on IoT DL UAV systems for autonomous obstacle detection and collision avoidance is presented in [58].

5.2. Spiking Neural Networks

Spiking neural networks (SNNs) are simulations of the functions and structure of the biological nervous system and its spike-based communication protocol. Widely referred to as the *third generation of neural networks* [107], they constitute the existing computational model with the highest level of realism and have the potential to become the most efficient and fast neural computational framework, thanks to the way they encode and process information. Neurons from biological neural systems exchange information by receiving and sending spikes –short electrical pulses– whose amplitude remains almost unchanged during propagation. Since all pulses have a very similar appearance, information encoding actually takes place through the precise timing at which these spikes are generated. This phenomenon is, according to several studies [50][107][156], what makes biological neurons and SNNs, their computational counterpart, so powerful in terms of high-speed computations. This is why spiking neurons belonging to SNNs are commonly referred to as asynchronous event-based processing units with temporal dynamics. This section dives deeper into the fascinating world of spiking neural networks, their biological background, existing models, training strategies, encoding mechanisms and more.

5.2.1. Biological background

For the explanation of the biological background behind spiking neural networks, this work refers to the theoretical knowledge gathered by [61] and [140]. As illustrated in Figure 5.2, three functionally different parts can be distinguished in a biological neuron: the dendrites, the soma and the axon. The former are specialized for receiving spikes from other neuron cells that are connected to them.

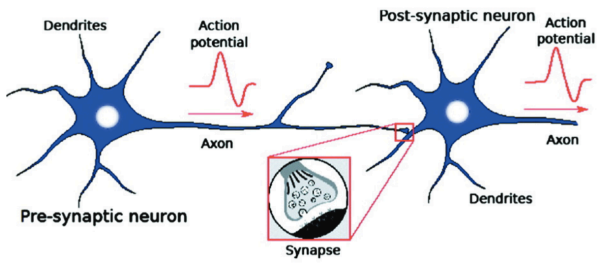


Figure 5.2: Biological neuron and synapse, from [80].

Then comes the soma, which is characterized by a state variable known as the *membrane potential* and acts as the processing unit of the neuron. Hence, if the excitation caused by the input spikes is strong enough so that the membrane potential reaches a certain threshold, the soma is in charge of triggering a spike. This short electrical pulse is propagated to other neurons via the axon. Finally, the actual information exchange between two neuron cells takes place at a *synapse*, which is a specialized structure that links two neurons together.

A synaptic transmission between the pre and postsynaptic neurons can be either excitatory or inhibitory depending on the kind of transmitting synapse. Concretely, when a train of incoming spikes from the presynaptic neurons causes a positive variation of the membrane potential, $u(t)$, of the postsynaptic cell, the synapse is considered excitatory. This type of synaptic transmission increases the likelihood of the postsynaptic neuron triggering an action potential after the stimulation. On the other hand, when the change is negative, it is said to be inhibitory, producing the opposite effect. The efficacy of a synapse –the strength of the postsynaptic response– is not fixed. The decrease or increase of the efficacy of a synapse is referred to as the *synaptic plasticity* and it enables the brain to memorize and learn. After its variation, $u(t)$ decays back to the resting potential u_{rest}. Finally, after the emission of a spike, the neuron enters in a short refractory period –typically, of some milliseconds– in which the membrane potential remains almost unchanged due to new incoming spikes. This ensures the event-based representation of the spikes as pulses that are clearly distributed over time.

5.2.2. Models of spiking neurons

Numerous mathematical abstractions of biological neurons have been inspired by the extraordinary information processing capabilities of the brain (see Figure 5.3). As already mentioned, spiking neurons account for the concepts of time, neural and synaptic state explicitly into the model [107].

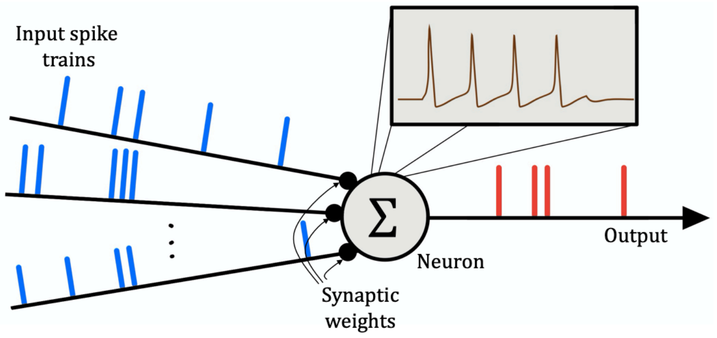


Figure 5.3: Diagram of the mathematical neuronal model. The efficacy of a synapse is modeled in the form of synaptic weights. Modified from [140].

Two distinct levels of abstraction can be considered within neural modeling:

1. The microscopic level, in which the neuron is modeled by describing the flow of ions through the channel of the membrane. The *Hodgin-Huxley Model* [77] is the most relevant example belonging to this category. This model proposes a set of differential equations capable of delineating the dynamics of a neuron and proposes, for the first time ever, that this dynamics can be modeled by means of electric circuits made of capacitors and resistors. The complexity inherent to this model, however, complicates its usage in large simulations of spiking neurons.

2. The macroscopic or *integrate-and-fire* models, which treat the neuron as a homogeneous unit that receives and emits spikes based on some defined internal dynamics. In the upcoming sections, special focus will be put on these type of models due to their relevance for the present work.

Leaky Integrate-and-Fire model (LIF)
The leaky integrate-and-fire model may be the most well-known model for simulating SNNs efficiently. Long before the actual mechanisms of action potential generation were understood, the model was first proposed by Lapicque in 1907 [100]. However, the first person who actually introduced the term integrate-and-fire was *Knight*, in [96]. He referred to these models as *forgetful*, although the term *leaky* quickly gained more popularity. Additional discussions on this work can be found in [5] and [26].

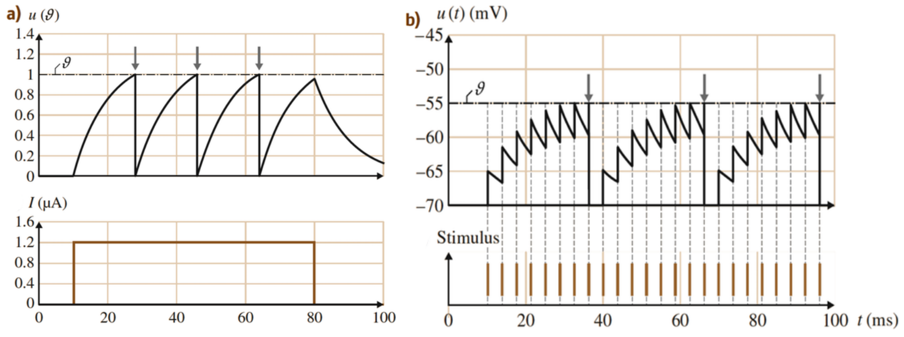


Figure 5.4: (a) Evolution of the potential u for a constant input current I_0 using the LIF model. As soon as I_0 vanishes, $u \to u_{rest}$. (b) Dynamics of the LIF model due to a train of presynaptic input spikes. Note that the synapse may be either inhibitory or excitatory, depending on the sign of the synaptic weights, w_{ij}. Modified from [140].

The LIF model, similarly to the Hogkin-Huxley model, is built upon the idea of an electrical circuit with a constant capacitor, C, and resistor, R, placed in parallel. This model assumes that the input channels to the neuron are static and that the shape of the presynaptic train can be neglected, so that only the firing times are considered relevant. The standard form of the model is shown in Equation 5.2.

$$\tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = -u(t) + RI(t), \tag{5.2}$$

where $\tau_m = RC$ denotes the membrane time constant. Whenever the membrane potential, u, reaches a certain threshold, ϑ, the neuron triggers a spike and u is reset to a resting potential, u_{rest} (see Figure

5.4). When a LIF neuron is part of a neural network, it is normally stimulated by the activity of its presynaptic neurons, which results in the synaptic input current illustrated in Equation 5.3.

$$I(t) = I_{syn_i}(t) = \sum_j w_{ij} \sum_f \alpha(t - t_j^{(f)}) \tag{5.3}$$

Therefore, the input current of a neuron i corresponds to the weighted sum of over all spikes triggered by the presynaptic neurons j with firing times $t_j^{(f)}$. The efficacy of the synapse from j to i is reflected on the weights w_{ij}. Finally, the time course of the postsynaptic current, $\alpha(\cdot)$ can be defined in many different ways: the simplest corresponding to a Dirac pulse, $\delta(x)$. Additional very interesting information concerning the LIF model can be found in [61], [28] and [27].

Izhikevich Model

This model, proposed by Izhikevich [85], claims to offer the computational simplicity of the LIF models while showing the biological feasibility inherent to the Hodgin-Huxley alternative. It reproduces different spiking behaviors of cortical neurons, by means of the dynamical expressions in Equation 5.4.

$$\begin{aligned} \frac{\mathrm{d}v}{\mathrm{d}t} &= 0.04v^2 + 5v + 140 - u + I \\ \frac{\mathrm{d}u}{\mathrm{d}t} &= a(bv - u), \end{aligned} \longrightarrow \text{if } v \geq 30\,mV \Rightarrow \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{5.4}$$

where v denotes the membrane potential and u corresponds to a membrane recovery variable. a, b, c and d are the model parameters and allow to simulate a myriad of neural characteristics. Concretely, a is the decay rate of the membrane potential, b the sensitivity of the membrane recovery, c resets v and d resets u. More information on this model can be found in [87], [86] and [88].

Spike Response Model (SRM)

The SRM [61], which corresponds to a generalization of the LIF model, describes the state of the membrane potential as an integral over the presynaptic spikes received in the past, without using differential equations (see Figure 5.5a). Concretely, the state of a neuron is characterized by a single variable u.

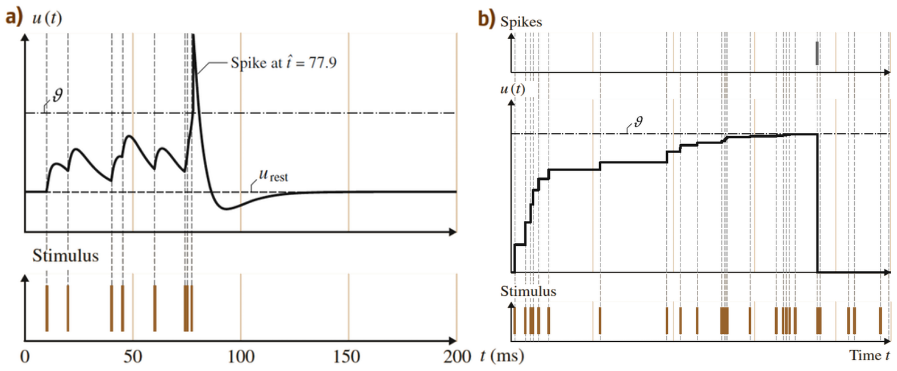


Figure 5.5: (a) Dynamics of the spike response model (SRM). (b) Evolution of the postsynaptic potential (PSP) of the Thorpe model. Modified from [140].

The impact of the presynaptic spikes, the shape of the actual postsynaptic spike and its after-potential, and the external simulation u are described by means of several kernel functions. Specifically, a spike is fired whenever the state u reaches a certain threshold from below: this is, $u(t) = \vartheta$ and $\frac{\mathrm{d}u(t)}{\mathrm{d}t} > 0$. In this case, however, ϑ may vary with respect to the last firing time $\hat{t}_i$ of neuron i; so it does not necessarily have to be constant. For example, to avoid firing another spiking right after one has been triggered (in what is known as the *refractory period*), ϑ may be increased. The evolution of $u_i(t)$ can be expressed as

$$u_i(t) = \eta(t - \hat{t}_i) + \sum_j w_{ij} \sum_f \epsilon_{ij}(t - \hat{t}_i, t - t_j^{(f)}) + \int_0^\infty \kappa(t - \hat{t}_i, s) I_{ext}(t - s) ds, \tag{5.5}$$

where $u_i(t)$ corresponds to the state variable that describes neuron i at time t, $\hat{t}_i$ denotes the last time when the neuron triggered a spike, w_{ij} represents the synaptic efficacy between neuron j and i (weights), $t_j^{(f)}$ are the firing times of presynaptic neurons j and, finally, η, ϵ and κ represent the response kernel functions. The reset kernel, η, delineates the dynamics of an action potential and becomes non-null each time a neuron fires. Then, ϵ determines the time evolution of a post-synaptic potential when the neuron receives an incoming spike. Finally, κ defines the linear response of the membrane to an input current I_{ext}. Further information about the SRM can be found in [61] and [108].

Thorpe Model
The Thorpe Model [154] is a simplified version of the LIF model which is characterized by lacking the post-synaptic potential leakage (see Figure 5.5b). Therefore, a neuron's spike response is only dependent on the arrival time of the presynaptic spikes. Additionally, the relevance of early spikes is enhanced, so that these have a higher impact on the postsynaptic potential than later spikes. This information processing mechanism makes possible that few spikes per neuron are biologically sufficient to solve a complex task in real time and, therefore, also simulations of large networks. Multiple works have investigated the applications of the Thorpe Model, for example, in the field of image, speech and face recognition [40][157][163][41].

5.2.3. Computational cost comparison between different spiking neurons

Since more than a decade ago, several studies on the accuracy and computational cost of different spiking neuron implementations have been published [87][146], leading to notably different results. This controversy points out that there is no clear consensus about the implementation capacities of distinct neuron models and reveals the need for more accurate methodologies to understand such capacities. In this way, the recently proposed work by Valadez-Godínez [161] attempts to shed some light on the matter, by performing a very detailed comparative study based on multiobjective optimization theory. The results obtained by each of these three comparative works are presented on this section both from a qualitative and quantitative perspective, in order to facilitate the task of choosing one or another model.

Izhikevich [87]	Skocik [146]	Valadez-Godínez [161]
1) The model by Hodgin and Huxley (HH) is prohibitive computationally	1) HHT model (HH with tables) is not prohibitive	1) HH model is the most efficient, computationally inexpensive and accurate
2) Izhikevich (IZH) model is as efficient as Leaky-Integrate-and-Fire (LIF)	2) IZH is not efficient	2) IZH is the most expensive and inneficient
3) IZH is more efficient than HH	3) IZH and HHT are similar regarding their computational cost	3) IZH is not comparable in computational cost to HHT
		4) LIF and HHT are the most innacurate
		5) HH is more accurate and inexpensive than HHT

Table 5.1: Conclusions drawn by the studies [87], [146] and [161] on the qualitative comparison of the efficiency, accuracy and computational cost of the different neuron models.

First, Table 5.1 presents the qualitative conclusions drawn by each of the aforementioned studies. From the first line of the table, the notable discrepancies existing among the different comparison works are tangible. These discrepancies are mainly due to the different methodologies used in order to perform the comparative study for each case, which are summarized here:

- In [87] only the Forward Euler (FE) numerical method (NM) was used and no metric was defined to assess the accuracy. Moreover, the time steps to solve the neuron models were arbitrarily assigned. Finally, the computational cost was only evaluated through floating point operations per second (FLOPS).

- In the case of [146], a voltage and frequency error was also included. Additionally, several NMs and well-defined time steps were compared.
- In [161] an extensive analysis where several firing frequencies, NMs, time steps and metrics were compared was conducted. Concretely, these were considered:
  - Spike Coincidence Factor (SPC): this metric allows to determine whether the spike-timing in a testing simulation is better against that in a reference simulation. It corresponds to the number of coincident pulses minus the number of related correspondences divided by the total spikes in both simulations.
  - Voltage Coincidence Factor (VCF): this parameter indicates whether the voltage in a testing simulation is better against that in a reference one.
  - Computational Cost Factor (CCF): this metric compares the normalized level of computational cost between a testing and a benchmark simulation.
  - Global Performance Factor (GPF): this parameter corresponds to the simplest objective function in the theory of multiobjective optimization. Its mathematical expression corresponds to $GPF = \frac{1}{2}CCF + \frac{1}{4}SCF + \frac{1}{8}VCF$.
  - Numerical Methods (NM): the considered NM are *Forward Euler* (FE), *Fourth-order Runge-Kutta* (RK4) and *Exponential Euler* (EE).
  - Time steps (ms): 0.0001, 0.001, 0.01, 0.1 and 1.

5.2.4. Neural encoding

One of the fundamental unknowns in neuroscience is the problem of neuronal coding. This is, given that the central nervous system is composed of around one trillion (10^{12}) neurons organized in very intricate networks [50], how do all these cells communicate between each other? Concretely, it would be interesting to know what the information contained in the spatio-temporal spike patterns is, how neurons encode and decode this information and, finally, how we, as observers, can understand these neural codes to decipher the message hidden in the neuronal activity pattern. Currently, even though there is no absolute response to all these questions, several useful strategies exist that can help to answer them partially. In this section, the three most important tools to decrypt the neural language will be analyzed: rate codes, pulse codes and rank order population encoding.

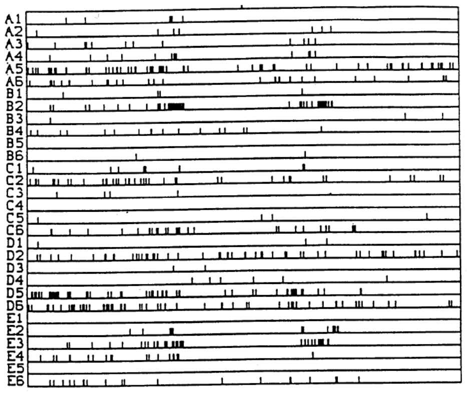


Figure 5.6: The spatio-temporal pulse patterns of 30 different neurons (A1 - E6), by [99].

Rate codes

Traditionally, it was assumed that the mean firing rate of a neuron is what carries most of the information of a transmission. There is, however, no universal definition for the concept of *mean firing rate*. Specifically, three different averaging procedures are considered:

1. *Rate as a spike count*: the first case corresponds to an average over time. The mean firing rate, v, is defined as the ratio of the average number of spikes, n_{sp}, observed over a predefined time interval T, so that $v = n_{sp}/T$. This approach has, however, several shortcomings. Firstly, it

neglects all the information that is possibly contained in the exact timing of spikes. Secondly, the processing time would be too high when, in reality, the brain is able to react to external stimulus in just a couple milliseconds. Hence, this practice could work well in cases where the stimulus varies slowly and does not require very fast reaction times.

2. *Rate as a spike density*: this alternative refers to an spiking average of a particular neuron over several runs of the same experiment. This approach is very useful when there is a large population of N neurons, so that it is not necessary to measure the outputs of every single unit, but just those of one unit over N runs of an experiment. However, it is not a realistic approach since, the decisions of any neural biological system always have to be based on a single run, as it is not possible to go back in time after a stimulus has occured.

3. *Rate as a population activity*: finally, the average over several neurons can be computed. In this case, an idealized network in which each neuron in population n receives inputs from all neurons in population m is considered. Also, all the neurons belonging to a certain population are assumed to have identical properties. As stated in [61], the relevant quantity for each receiving neuron in n is the proportion of active presynaptic neurons in population m. Most of the times, however, populations are inhomogeneous so that the assumptions made are not valid. If this is the case, a weighted average over the population should be calculated instead.

Pulse codes
As opposed to rate codes, pulse codes do assume the precise spike time as the information carrier among neurons. The most popular pulse code strategy is called *time-to-first-spike*. As its name suggests, this idealized approach assumes that, for each neuron, the timing of the *first* spike after the reference signal contains all the information about the new stimulus. Therefore, only one spike per neuron would suffice to transmit information according to this scheme, so that the number of spikes would be irrelevant. Even though the time-to-first-spike seems to be an idealization, some studies have argued that the brain does not have enough time to process more than one spike per neuron on each step [155].

Rank order coding
A different alternative is to just pay attention to the order in which the neurons fire, rather than the precise timing of spikes [154]. The main advantage of rank order coding is its high simplicity and lower computation time when dealing with large networks. An example of this coding mechanism is illustrated in Figure 5.7a, where the neurons could be thought as transmitting the order $C > B > D > A > E$. This corresponds to just one of the $5!$ orders that can be obtained with 5 neurons.

Rank order population encoding
First proposed by Bohte in [20], the rank order population encoding strategy allows to map vectors of real-valued elements into a sequence of spikes. This is something very useful when an event-based sensor is not available to take measurements, which is the case of this work: the radar sensor used corresponds to a frame-based sensor. It could be thought that this encoding mechanism has some elements from both pulse and rank order codes. Concretely, [20] states:

> *[...] we employed an encoding based on arrays of receptive fields. This enables the representation of continuously valued input variables by a population of neurons with graded and overlapping sensitivity profiles, such as Gaussian activation functions (the receptive fields, RFs). To encode values into a temporal pattern, it is sufficient to associate highly stimulated neurons with early firing times and less stimulated neurons with later (or no) firing times, and then the extension to temporal coding is straightforward.*

Figure 5.7b illustrates the working principle behind this encoding strategy. For more information on the mathematical details of this approach, please refer to [20]. Additional interesting information about rank order population encoding can be found in [41], [125], [156] and [106].

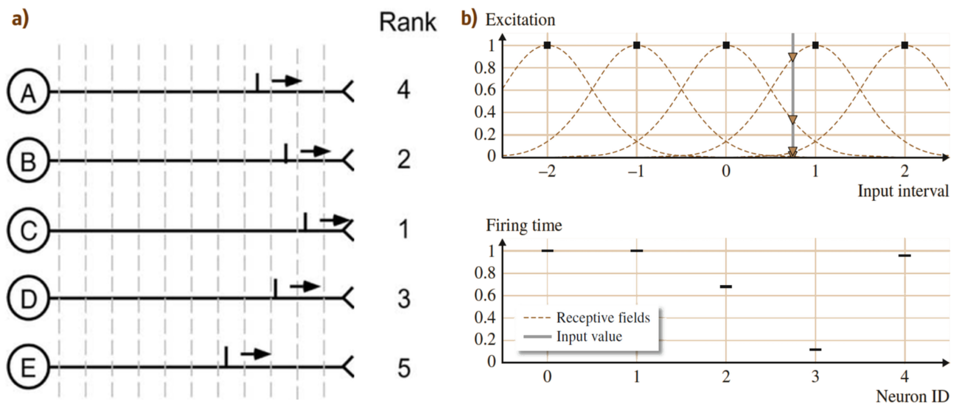


Figure 5.7: (a) Diagram representing the working principle of rank order coding. Modified from [156]. (b) Diagram illustrating the logic behind rank order population encoding. Modified from [140].

5.2.5. Learning strategies: synaptic plasticity

According to [10], *synaptic plasticity* corresponds to "the basic mechanism underlying learning and memory in biological neural networks". In the realm of neuromorphic computing, it could be understood as the increase or decrease of the synaptic connections strength of a neural architecture. This synaptic strength, also referred to as the *efficacy* of a synapse, can be represented by the weights w_{ij} already seen in the Spike Response Model (see Equation 5.5).

Similarly to traditional neural networks, three different learning criterion can be discriminated when it comes to SNN: unsupervised and supervised learning, and reinforcement learning. Nevertheless, in the case of SNN, their inherent time dependency, asynchronous information processing and commonly used recurrent network topologies impair the development of straightforward learning procedures, such as backpropagation for MLP. This poses a great additional challenge that is still being investigated.

Unsupervised learning

All the existing unsupervised learning methodologies, based on inducing changes in the efficacies of the synapse, are based on the following postulate, by [45]:

When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.

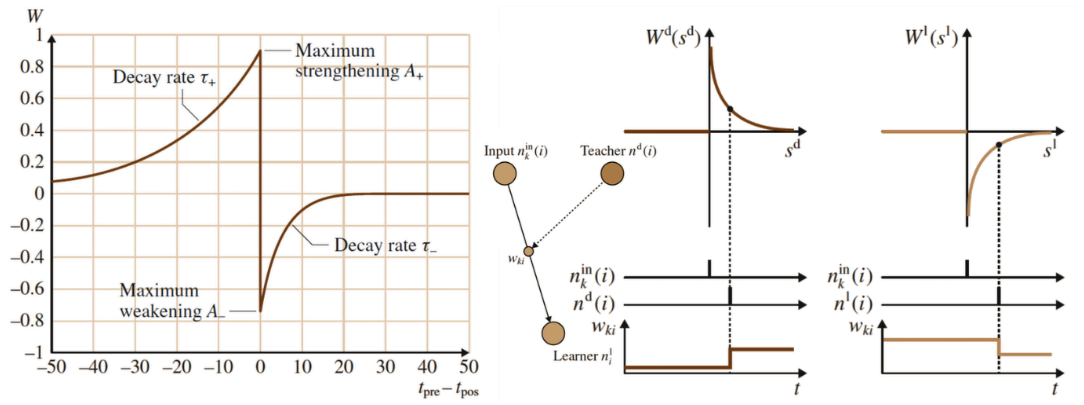


Figure 5.8: (a) Plasticity window characteristic of the STDP rule. (b) Illustration of the remote supervised method (ReSuMe). Images from [140].

This postulate, known as Hebb's Law, was first published in Hebb's famous book *The Organization of Behaviour*, and is the main reason why unsupervised learning in the context of SNN is widely referred to as *Hebbian learning*.

Spiking-Timing Dependent Plasticity (STDP) [111] is undoubtedly the most famous Hebbian learning strategy. Its working principle its based on the correlation that exists between the synaptic efficacy and the timing of the pre and postactivity of a neuron [16]. Concretely, whenever the presynaptic pulse arrives to the postsynaptic neuron before it triggers a spike ($\Delta t = t_{pre} - t_{post} > 0$), the connection is strengthened. Otherwise, it is weakened. The STDP window function, $W(\Delta t) = W(t_{pre} - t_{post})$, describes the exact fractional change of the synaptic efficacy or synaptic weight. This is reflected in Equation 5.6:

$$W\left(t_{pre} - t_{post}\right) = \begin{cases} A_{+} \exp\left(\dfrac{t_{pre} - t_{post}}{\tau_{+}}\right) & \text{if } t_{pre} < t_{post}, \\ A_{-} \exp\left(-\dfrac{t_{pre} - t_{post}}{\tau_{-}}\right) & \text{if } t_{pre} > t_{post}, \end{cases} \tag{5.6}$$

where parameters $\tau_{\pm}$ delineate the temporal range of the pre and postsynaptical time interval, respectively, while $A_{\pm}$ determine the maximum fractions of synaptic weight modification. All these parameters must be adjusted depending on the neuron that is to be modeled. Additional information on STDP learning can be found in [15], [92], [60] and [95].

Supervised learning

To the best of the author's knowledge, the work by Wang et al. [167] comprises, by far, the most recent and comprehensive review of the state-of-the-art supervised learning algorithms that have been applied to SNN learning to this day. According to this work, the supervised learning algorithms for SNNs developed in recent years, can be divided into different categories:

$$\begin{array}{c}\text{supervised learning}\\ \text{algorithms for SNNs}\end{array} \begin{cases} \text{network architecture} \begin{cases} \text{single layer} \\ \text{multilayer feed-forward} \\ \text{recurrent} \end{cases} \\ \text{running mode} \begin{cases} \text{online learning} \\ \text{offline learning} \end{cases} \\ \text{information encoding} \begin{cases} \text{single-spike learning} \\ \text{spike train learning} \end{cases} \\ \text{structural dynamics} \begin{cases} \text{learning in fixed SNN structures} \\ \text{learning in evolving SNN structures} \end{cases} \\ \text{knowledge representation} \begin{cases} \text{non-knowledge-based learning} \\ \text{learning for knowledge representation} \end{cases} \end{cases}$$

Figure 5.9: SNNs supervised learning algorithms classification. Image from [167].

Given the large number of existing supervised learning algorithms for SNN learning depending on the desired approach (see Figure 5.9), it would be impossible to treat them all. Therefore, this work focuses on some of the most relevant ones: spike-prop, liquid state machine (LSM) and remote supervised learning (ReSuMe).

First, a back-propagation algorithm named spike-prop, which is derived from the SRM already discussed, is suggested in [21] and [20]. Its objective is to learn a collection of desired firing times, t_j^d, of all output neurons j for a certain input pattern presented to the network. Equation 5.7 shows the definition of the error function that is to be minimized (left) with respect to the weights w_{ij}^k of each synpatic input (right):

$$E = \frac{1}{2}\sum_{j}\left(t_j^{out} - t_j^d\right)^2, \quad \Delta w_{ij}^k = -\eta \frac{\mathrm{d}E}{\mathrm{d}w_{ij}^k}, \tag{5.7}$$

where t_j^{out} corresponds to all the obtained network firing times and η is the learning rate of the updated step. The spike-prop algorithm only allows each neuron to fire one spike and is only compatible with a time-to-first-spike encoding scheme. Several interesting modifications have been introduced in [89], [142] and [158].

A notably different algorithm named the liquid state machine (LSM) was introduced in [109]. This mechanism builds a recurrent SNN where all the network parameters are randomly selected and kept constant during the simulation, in what is known as the *liquid*. Then, a supervised learning algorithm is applied to a set of training samples $(x(t), v(t))$ in order to train a *readout* function, f, such that $f(x(t))$ are as close as possible to $v(t)$. The most compelling feature of this strategy is its inherent simplicity, since the readout function corresponds to a single layer of weights only, so that a linear training mechanism suffices.

The remote supervised method (ReSuMe) proposed in [127], whose goal is to impose a desired input-output spike pattern on a SNN, corresponds to a particular implementation of the aforementioned readout function. Its aim is to generate target spike trains as a reaction to some input stimulus. This methodology is based on the learning window function of the STDP already explained, and its working principle can be observed in Figure 5.8b. In this case, apart from the presynaptic neuron (*input* $n_k^{in}(i)$) and the postsynaptic one (*learner* n_i^l), an additional neuron, the *teacher* $n^d(i)$, that is not explicitly connected to the network but remotely supervises the evolution on the weights w_{ki}, is defined for each synapse. Concretely, the synaptic weight, w_{ki}, increases whenever the input neuron spikes before than the teacher does. When the input neuron fires before the learner is activated, w_{ki} decreases. Finally, functions $W^d(s^d)$ and $W^l(s^l)$ define the amplitude of the synaptic change. Additional information on ReSuMe can be found in [127], [128] and [91].

5.2.6. Learning strategies: evolution

The inherent nonlinear and discontinuous mechanisms of SNNs, highly complicate the formulation of efficient learning algorithms [167]. To address this problem, neuroevolution strategies inspired by the Darwinian evolutionary laws pose an auspicious alternative for SNN learning [53].

Evolution starts with a population of organisms/agents, and some way of representing their genome. A way to represent their genome is needed, as the evolution process requires variation; this is, creating offspring that have a chance of being different from their parents. The way the offspring is variated is by applying variation operators to their parents' genomes. For example, by first cloning the parent genome and then mutating that cloned genome. Another important element that is required is the process of selection. In biological evolution, the fittest organisms are those who create the most offspring. Therefore, the ability to create offspring is that which defines the organism's fitness [144]. The two main operators used to generate the offspring are *mutation* and *crossover*, as showcased in Figure 5.10.

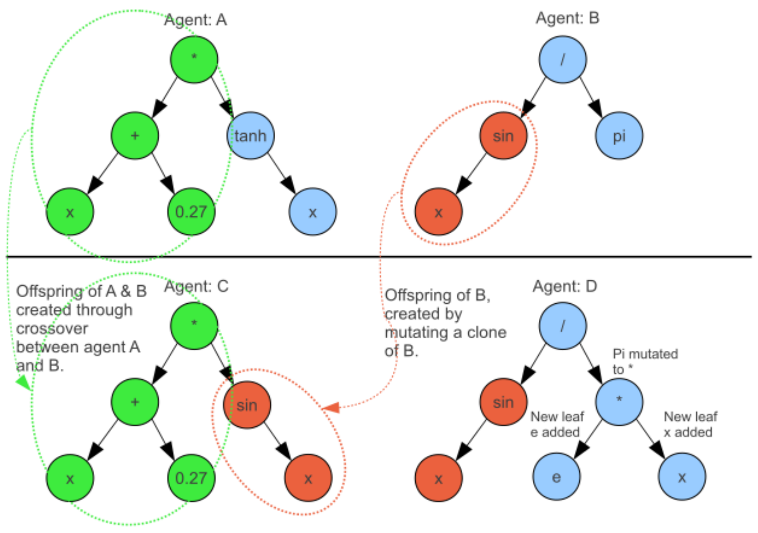


Figure 5.10: Offspring generation through mutation and crossover. Image from [144].

Some interesting applications of evolutionary training strategies in the real of UAV control include: a spiking thrust controller which successfully conducts the landing of a MAV by means of the optical flow divergence from a down-ward looking camera [69]. This constitutes one of the two existing works that integrate the SNN controller in a real flying platform, as far as we know. In [46], the previous work is extended and tested on a neuromorphic processor. Some other evolved SNN controllers have been applied to MAVs [78][79], but only in simulation.

5.2.7. SNN and their vision- and control-based applications

Learning real-world robot control can be characterized as the optimization of some kind of behavioral function in an uncertain, complex environment. Reinforcement learning (RL) in combination with ANN has been one of the most widely used approaches when it comes to solving this type of problems [81]. However, despite the many advantages that, as already discussed, ANN offer, they also pose several disadvantages. For example, their training is very time consuming and can even take up to several days for state-of-the-art techniques. Performing calculations with large networks can also become computationally expensive. Therefore, they require powerful hardware that is incompatible with small flying robots such as MAVs or blimps, which is the case of this work. The same applies when it comes to object detection or tracking, image classification and image segmentation tasks. Thus, in this section, some interesting applications of SNN both in the field of vision and robot control are analyzed, together with the potential they offer.

Applications of SNN in the realm of vision

Before diving into the details of the multiple applications that SNN have recently shown in the field of artificial vision perception, the two most important types of existing vision sensors must be analysed:

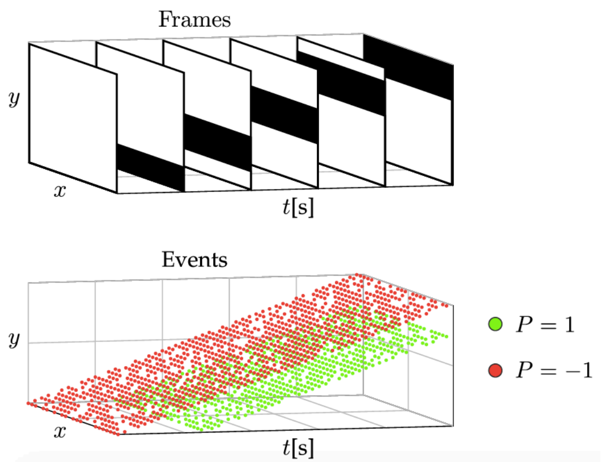


Figure 5.11: Contrast between the output of a frame- and event-based sensor under the stimulus of a black bar moving upwards over a white background, by [123].

- First, there is the category of *frame-based* sensors, which comprise the working principle of most of the cameras utilized for visual perception. In this case, data is gathered by measuring the brightness levels of a pixel array at fixed time intervals. Also, the frame rate is uncorrelated from the dynamics of the visual scene.

- In the case of *event-based* sensors (e.g. [103]), each of the pixels in the array reacts to brightness changes in an asynchronous manner in its corresponding receptive field by generating events. The great precision of these sensors and their low power consumption make them ideal candidates for visual perception tasks. Figure 5.11 illustrates the working principle of the each of these two categories.

Due to the synchronous nature inherent to ANNs and asynchronous discrete spiking communication protocol linked to SNN, ANNs have been widely used in conjunction with frame-based strategies and SNNs have shown their great potential with event-based sensors. However, none of these two neural network approaches are restricted to a particular type of sensor.

Objective	Neuron	Learning	Encoding	Work
Learning visual features	IF	STDP	Time-to-first-spike	[112]
Digit recognition	LIF	STDP	Poisson spike train	[44]
Object recognition	IF	STDP	DoG (latency)	[94]
Object recognition	LIF	STDP	-	[153]

Table 5.2: Examples of image classification based on the STDP learning algorithm.

The dominant learning algorithm in the realm of spike-based image classification is the Spike-Timing Dependent Plasticity [95] already presented. This algorithm has been successfully utilized for relatively easy image classification tasks in [112], [44], [94], [153] or [145]. Further details about these implementation can be found in Table 5.2. SNN-based strategies have also been successfully implemented in several image segmentation tasks. Some examples include: edge detection [173], color segmentation [174], corner detection [93], visual attention [172], all built upon a similar conductance based IF neurons architecture. A different architecture based on the SRM neuron model and Gaussian receptive fields to convert real values into spikes was used for edge detection and color segmentation in [114].

Robotics control based on learning-inspired SNN

Even though reinforcement learning has successfully been applied in combination with ANN to a myriad of robot control tasks, it is still an immature strategy when it comes to SNN, as it has only been applied to a few number of cases. RL inspired approaches to SNN learning are commonly referred to as *reward modulated* as the training strategy conflates models of STDP and a global reward signal [54]. The synaptic weight w changes with the reward R in the R-STDP rule. The elegibility trace of a synapse is defined as,

$$\dot{c}(T) = -\frac{c}{\tau_c} + \omega(\Delta t)\delta(t - s_{pre/post}C_1) \tag{5.8}$$

where c represents the elegibility trace, $s_{pre/post}$ the time of pre or postsynaptic spikes, C_1 a constant coefficient, τ_c a time constant of the elegibility trace and $\delta()$ the Diract delta function. Moreover, $\dot{w}(t) = R(t) \times c(t)$, with $R(t)$ the reward signal. A large variety of algorithms that use this basic learning structure for learning have been published in the literature. Even though the underlying mechanism is the same for all of them, the reward can be constructed in several ways: reward specific events [48], control error minimization (optimization task to minimize an objective function) [35], indirect control error minimization [55], metric minimization [32] and reinforcement associations [34]. Some of the works that have recently opted for a combination of RL and SNN include: an extremely simplified version of an obstacle avoidance task on a real-world MAV [177], a vision-based neuromorphic lane-keeping control of a two-wheeled small robot [19] and a simulated robot insect simplified control task [35]. A more detailed list is presented in Table 5.3.

Another extensively used approach lies on the field of neuroevolution. However, most of the successful evolved SNN controllers for MAVs have only been achieved in simulated environments [78][79] or simplistic versions of real-world ground robots [17], except for [69], which manages to integrate the evolving SNNs in the control loop of a real-world flying robot.

Classical conditioning learning procedures have also been applied to SNN. This kind of learning pairs a biologically potent stimulus such as food (*unconditioned stimulus, US*) with a previously neutral stimulus like a bell (*conditioned stimulus, CS*). In the end, the neutral stimulus will be able to elicit a response such as salivation which was previously elicited only by the potent stimulus. Using classical conditioning for robot control implies building an external controller that provides US for every relevant state input; something that may not always be feasible.

Some of the most important strategies of robotics control based on learning-inspired spiking neural networks developed in recent years are illustrated in Table 5.3.

Learning	Rule	Neuron	Robot	Objective	Reference
Unsupervised	STDP	LIF	Two-wheel vehicle	Navigation tasks	[136]
Unsupervised	STDP	IF	Mobile vehicle Casis-I	Navigation tasks	[168]
Unsupervised	STDP	LIF	Two-wheel insect	Navigation and avoidance	[43], [113]
Unsupervised	STDP	HH	Aircraft	Approximate flight controller	[55]
Unsupervised	STDP	Izhik	4-DoF robotic arm	Position control	[22]
Unsupervised	STDP	IF	2-DoF robotic arm	Position control	[31]
Supervised	LSM	Izhik	Hexapod robot	Recurrent SNN modeling	[7]
Supervised	LSM	LIF	2-DoF platform	Position control	[129]
Supervised	LSM	LIF	Khepera robot	Obstacle avoidance	[18]
Conditioning	STDP	LIF	Lego NXT 2.0	OC learning	[36]
Conditioning	STDP	IF/Izhik	Simulated fly	Flight simulation	[48]
Conditioning	STDP	HH	Mobile vehicle Casis-I	Path planning	[166]
Conditioning	STDP	Izhik	TriBot robot	Navigation and avoidance	[8]
Reinforcement	R-STDP	LIF	Flapping insect	Adaptive flight control	[35]
Reinforcement	R-STDP	Izhik	1-DoF robotic arm	Target reaching task	[149]
Reinforcement	R-STDP	-	Foraging simulator	Grid-based foraging task	[148]
Reinforcement	R-STDP	-	1-DoF robotic arm	Target reaching task	[32]
Reinforcement	R-STDP	Izhik	Two-wheel vehicle	Food foraging task	[47]
Evolutionary	GA	Izhik	Neural Racing game	SNN-MLP comparison	[110]
Evolutionary	GA	LIF	Quadrotor	Weight combinations SNN	[78]
Evolutionary	GA	SRM	Two-wheel vehicle	Control heuristic rule	[9]
Evolutionary	GA	SRM	Two-wheel vehicle	Navigation task	[52], [51]
Evolutionary	GA	SRM	Two-wheel vehicle	Obstacle avoidance	[6]

Table 5.3: Different SNN learning rules applications. Modified from [17].

5.2.8. Simulators and hardware implementations

For the development of novel control strategies based on complex multi-layer SNN, fast processing simulation environments of these networks, that can handle different types of learning rules and models, are of great importance. Luckily, the notable growth of SNN applications in neuroscience and engineering have also given rise to many interesting simulation frameworks in recent years. First, to design and simulate empirically-based models of biological neurons the NEURON[1] [73] and GENESIS[2] [24] framework exist. These environments, however, are not very suitable for large network simulations where the focus is put on the actual interaction between neurons and not the model itself. For this purpose, NEST[3] [62], Brian[4] [65] or DAMNED [117] are available. Several libraries that have been re-

[1]https://neuron.yale.edu/neuron/
[2]http://www.genesis-sim.org/GENESIS/
[3]https://www.nest-simulator.org/
[4]https://briansimulator.org/

cently developed also include: SpykeTorch[5] [118], a PyTorch-based simulator of convolutional spiking neural networks, in which the neurons emit at most one spike per stimulus; BindsNET[6] [71], a package used for simulating SNN on CPUs or GPUs using PyTorch Tensor functionality; or PySNN[7], a low-level framework written on top of PyTorch for efficient simulation of SNNs both on CPU and GPU. Additional information about tools and strategies for the simulation of SNN can by found in [25].

To accelerate the computation of SNN, several new hardware platforms (neuromorphic ASICs and ASIPs) have been lately developed simultaneously to the discussed simulation frameworks counterparts. The most relevant ones comprise TrueNorth from IBM [115], SpiNNaker from the University of Manchester [59], ROLLS from INI Zurich [130], Neurogrid from Stanford University [13], Loihi from Intel [37] and BrainScaleS from European Consortium [138]. More information about state-of-the-art hardware implementation of SNN can be found in the comprehensive survey [23]. It is worthwhile to specially highlight Imec's recent creation: the first recurrent spiking neural network-based chip for radar signal processing [84].

5.3. Comparison between ANNs and SNNs

Finally, a comparison between the newer spiking neural networks (SNNs) and the traditional artificial neural networks (ANNs) is presented in Table 5.4. Concretely, the type of information encoding and representation, together with the kind computational unit and network simulation, the synaptic plasticity and learning and the parallel and hardware implementation are contrasted.

Characteristics	**SNN**	**Traditional ANN**
Encoding scheme	Temporal encoding	Rate encoding
Information representation	Spike trains	Scalars
Dealing with (spatio)temporal data	Excellent	Moderate
Neuron model	Spiking neuron	Artificial neuron
Computation mode	Differential equations	Activation function
Network simulation	Clock- and event-driven	Step-by-step
Plasticity mechanism	STDP rule	Hebb rule
Designing SL algorithms	Various mentalities	Loss f. derivative
Parallel computation	Massive	Moderate
Hardware support	Neuromorphic VLSI	VLSI

Table 5.4: Comparison between SNN and traditional ANN, from [167].

[5] https://github.com/miladmozafari/SpykeTorch
[6] https://github.com/BindsNET/bindsnet
[7] https://github.com/BasBuller/PySNN

III

Neuromorphic altitude controller development and evaluation

6 Introduction to the methodology

The methodology and scope of the project can be divided into seven main parts: the review of the existing literature, the design of the physical blimp platform, the selection and integration of the actuators and embedded systems into this platform, the signal processing and integration of the radar sensor, the training of the controllers in simulation the transfer of these controllers to the actual blimp robot and their evaluation in a real-world environment and, finally, the validation of these results. As the project progresses, each of these steps will be analyzed and improved in an iterative process. In this way, the methodology and chronology of the project is summarized as follows:

1. **Literature review of the state-of-the-art**
   First, most of the state-of-the-art works concerning the five main parts of the project are reviewed. This step allows us to then select the most appropriate methods and computational tools available for our work. Moreover, it provides us with a clear idea of what already exists and the opportunities for new contributions. From this literature review we see that, even though most SNN-based controllers for flying robots have excelled in simulated environments, they have not been evaluated in a real-world setup. Furthermore, to the best of the author's knowledge, no works have quantitatively evaluated and compared the performance of an SNN-based controller with other more traditional approaches, such as an ANN or a PID, again, in reality. Finally, few works involving airships have worked with gondolas that are versatile and light-weight enough to be mounted on commercially available envelopes and, therefore, the total cost of the platform remains notably high.

2. **Airship design**
   The robot for the project has been designed from scratch. This involves everything from the selection of the envelope, all the electronic components, the design of an appropriate gondola frame, the low-level programming of every single pin and component to achieve an appropriate software integration, the development of a teleoperation package that would allow for an easy and intuitive manual control of the blimp, and the appropriate mechanical integration such that the airship would move around as desired, based on the software packages generated. Concretely, the design of the gondola follows an iterative process, until its versatility and weight reaches the desired value, without compromising its durability and strength, to allow for an appropriate control of the plant. Therefore, an initial gondola is developed taking inspiration in some additional works. Then, this designed is progressively improved until a satisfying final prototype is created. This is then 3D printed and attached to the envelope for further testing.

3. **Components and actuators selection, and their integration**
   The selection of the appropriate actuators for the project has mainly been done by taking inspiration in [49] and [66]. Moreover, the main focus has been to minimize the weight and size of the components as much as possible, while keeping the price low, and still allowing for a decently accurate control of the platform. Once all the components were selected, their integration has been performed within the Robot Operating System (ROS) [131] framework. Concretely, the `WiringPi` library is used to control the speed of the DC motors by means of a hardware PWM

signal using C++, and the `pigpio` library is used for the control of the servomotor angle via a software PWM signal, in Python.

4. **Radar sensor**
   The short-range Infineron Position2Go radar sensor is the only sensor that is mounted on the airship. At the time of starting the project, a range-Doppler algorithm that provides measurements of the range, speed and angle of detected targets is already fully functional. This algorithm, however, is implemented in OpenFrameworks and needs to be transferred to ROS. Moreover, an adequate filtering of the received signal is performed. For this, a Median Filter that takes care of removing possible outliers is implemented, together with several moving average (MA) filters that smooth the received signal. The performance of the latter alternative is evaluated on several sets of measurements to choose the option that best suits the requirements of our work.

5. **Training in a simulated environment**
   The training of the controllers takes place in simulation. Moreover, given the inherent time dependency and asynchronous information processing that is inherent to SNNs, no standard training algorithms such as the equivalent of backpropagation for ANNs have been yet developed. To solve this learning problem, we select an evolutionary training strategy to train the neurocontrollers offline. This approach applies the Darwinian evolutionary principles of mutation. Therefore, a random population of neurocontrollers (individuals) is generated, and their performance (fitness) evaluated on a certain set of altitudes. Then the parameters of a certain percentage of individuals are mutated over a number of generations and their performance is reassesed. The idea is that only the strongest individuals, this is, those with the best fitness, will survive over the generations. In this case, the fitness is quantified as the root mean squared error (RMSE) between the desired altitude that we wish to attain and the current altitude of the blimp. Furthermore, to perform these computations in simulation, a model of the airship is generated and added to the loop.

6. **Transfer to reality**
   Once the neurocontrollers have been evolved in simulation and we have selected the best performing individuals for both cases -ANN, and SNN-, these are transferred to the actual airship. Inevitably, every time that we migrate a controller from a simulated environment to the actual platform, a reality gap will exist, due to the mismatch between the working conditions in simulation and reality. For example, (1) the model of the airship is an approximation of the true blimp, which cannot perfectly account for its (nonlinear) dynamics; (2) the noise generated by the radar cannot be perfectly modeled computationally; (3) the temperature and amount of helium in the airship moderately vary over time, slightly affecting the amount of lift generated; (4) or the wind disturbances present in the testing arena, which are not accounted for in simulation. This implies that the results obtained in simulation will defer from those that are observed when testing the setup in the real-world environment. Therefore, we first perform the transfer crudely, to assess how large this difference is, without any additional tools that can help reduce this mismatch. At this point, we evaluate whether the airship can still follow the altitude setpoints marked. In this case, we observe that the blimp is able do this, but with the presence of some undesired oscillations. To minimize this effect, we propose a procedure to dampen the oscillations observed with the neurocontrollers, by adding a parallel PD controller tuned in reality which does not have a contribution larger than a certain value that we set. This parallel PD can partially take care of the reality gap thanks to the fact that it is directly tuned on-site, complementing the main neurocontroller which has been fully developed computationally.

7. **Validation of the results**
   To validate the results, a quantitative comparison among the SNN, ANN and PID controller is conducted. Mainly, the accuracy and control effort are contrasted. This is, how precisely the blimp can follow the altitude waypoints, together with how energy efficient each of the controllers are, in terms of the absolute magnitude of the motor command generated over time. To assess this, multiple test flights are performed in the CyberZoo flight arena of the MAVLab, at TU Delft. The desired and measured altitude (both with the OptiTrack motion capture system and the radar), together with the motor commands send to the airship are recorded and analyzed.

This chapter provided a overview of the project methodology, by briefly discussing each part of the work separately. In the upcoming chapters of Part III, a more detailed description of each of these parts is provided, to gain further insight into the followed methodology. Therefore, the reader that seeks for more in-depth explanations to those given in the present introduction or the scientific paper enclosed in Part I, is encouraged to move to the part of interest for further details.

Finally, a Gantt chart providing a view of the tasks to be completed scheduled over time is illustrated in Figure 1, in the Appendix.

7 Airship design and assembly

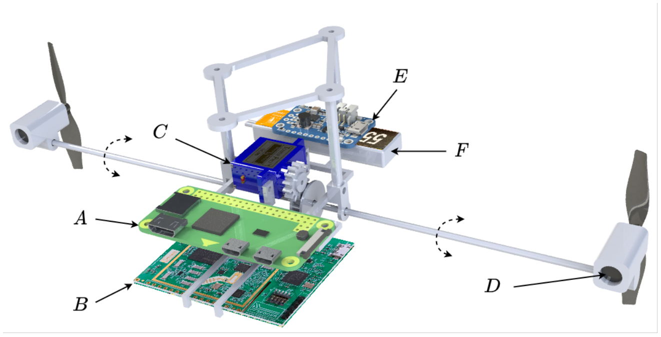
The development of a low-cost airship capable of operating autonomously in indoor environments is a fundamental piece of the puzzle for the present work. With the proposed design, we push forward the state-of-the-art by designing a 3D-printable, low-cost, versatile, light-weight and open-source blimp platform with a reduced number of actuators. The developed platform can be easily reproduced and adapted for different project applications by adding and/or modifying the on board sensors. Moreover, its ease of use and generalization properties make it ideal for robotics research.

Therefore, a detailed explanation of the blimp's design process is delineated in the present chapter. First, Section 7.1 describes the electronics used, together with their cost, the different parts of the gondola and their design, the total weight of the airship and the schematics of the final assembly. Then, Section 7.2 explains and contrasts the evolution between the first and final gondola prototypes. Finally, Section 7.3 focuses on the altitude ranging sensor specifically, by presenting its specifications and the signal processing algorithms utilized.

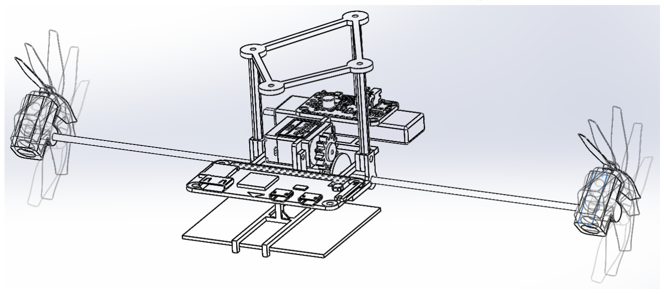
7.1. Blimp envelope and electronic components selection

7.1.1. Specifications and costs

The gondola can be 3D printed and is assembled in a modular fashion. Its open configuration makes it notably versatile, leaving room for the selection of custom actuators and sensors, which can be easily interchanged. Moreover, it merely uses one low-power micro servomotor and two coreless DC motors to achieve the three main motion primitives: forward, upward/downward, and yaw motion. The DC motors are accommodated on both ends of the rotary shaft, which is controlled by the servo. The rotation starts at 0°(upward movement) and can go up to 180°(downward movement). The thrust of each of the propellers can be controlled independently, to achieve the desired yaw. Finally, we attach the gondola to the airship by means of Velcro strips places on the top four hitches. Figures 7.1a and 7.1b present the gondola and its different components, and a simple scheme showcasing the effect of the shaft rotation on the global configuration, for better visualization, respectively.



(a) Gondola frame and electronic components used. The components corresponding to each letter reference are listed in Table 7.1.



(b) Skeleton view of the gondola, showcasing the rotary shaft mechanism.

Figure 7.1: Scheme of the proposed gondola design.

Table 7.1 presents the different components used and their cost. Adding up this prices and including that of the envelope, which is if $\sim \$20$, makes a total of $\sim \$70$ to build the entire flying robot. The radar is not taken into account in this calculation, as it can be easily replaced by some other more affordable sensors, depending on the application. Moreover, concerning the envelope's material we select Microfoil due to its high helium retention properties [66]. The model selected is the one that provides the largest achievable payload ($\sim 150g$) among the commercially available micro blimps, to the best of our knowledge, while maintaining a low price. Figure 7.2 shows the final airship configuration with the chosen envelope and the 3D printed gondola attached to it via three Velcro strips.

Reference	**Name**	**Quantity**	**Price** [$]
A	Raspberry Pi W Zero	1	10.00
B	Infineon Radar Position2Go	1	-
–	DRV8833 Motor Driver	1	4.95
C	Sub-micro Servo SG51R	1	5.95
D	8520 Coreless Motor	2	8.00
E	PowerBoost 500 Basic	1	9.95
F	550mA 3.8V Li-Po Battery	1	7.95

Table 7.1: Airship components (Figure 7.1)

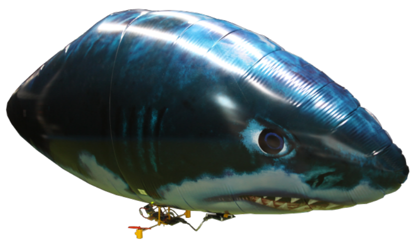


Figure 7.2: Blimp's Microfoil envelope with the gondola attached.

To perform the computations on board and integrate the different components, the Robot Operating

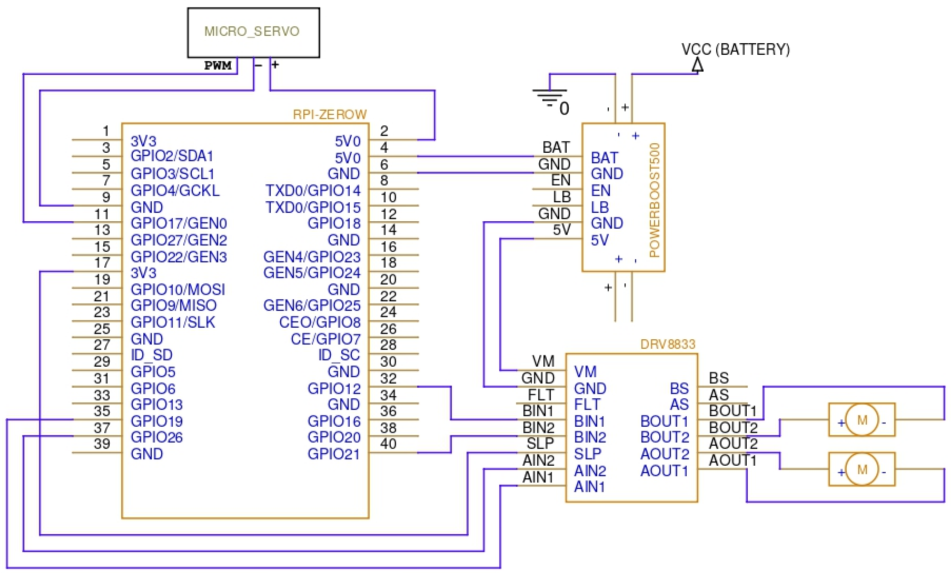
System (ROS) [131] framework is used. The Raspberry Pi Zero W, running the Raspbian Lite operating system, acts as the central control and communication unit. The blimp can be manually controlled from the ground computer through the provided teleoperation package. The total weight of the different parts of the platform is presented in Table 7.2.

Frame $[g]$	**Components** $[g]$	**Envelope** $[g]$	**Helium** $[g]$	**Total** $[g]$
9	64	54	20	147

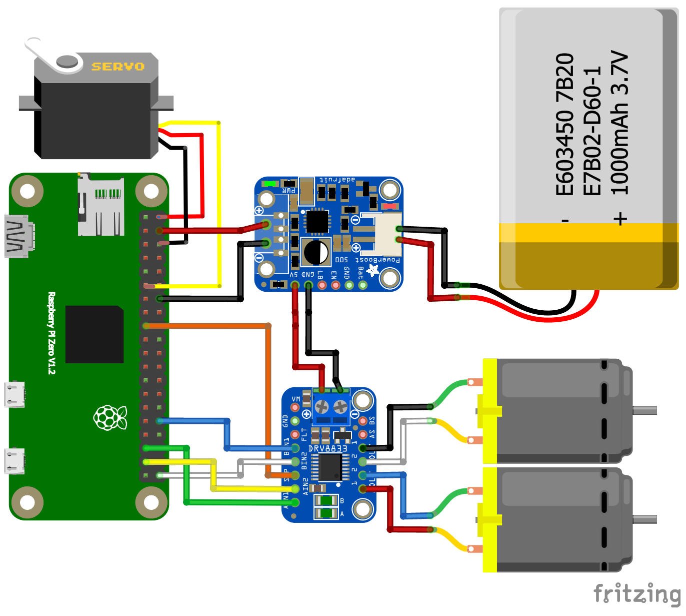
Table 7.2: Weights of the different airship's parts.

7.1.2. Assembly schematics

The sensor and electronic components shown in Table 7.1 are chosen to provide complete functionality while requiring low currents. The Li-Po battery is selected to be lightweight (~ 10g), with a capacity of 500 mAh, which allows for a total of $20-30$ min of operating flight time. The Infineon Radar Position2Go is chosen as the one and only sensor carried on-board, providing the altitude measurements (see Section 7.3 for more details).



(a) Abstracted electronic component's assembly schematics.



(b) Realistic scheme showing the connections among the different components.

Figure 7.3: Proposed assembly schematics.

Figure 7.3 shows the complete assembly schematics with all the different components and the way in which they are interconnected. Concretely, Figure 7.3a presents the connections and pin names in a detailed way, while Figure 7.3b displays the same scheme in a more realistic fashion, for better visualization. Following these pictures, we can see that the 500 mAh battery supplies current to the PowerBoost 500 Basic step-up voltage regulator. The latter feeds the Raspberry Pi Zero W at 5V and the DRV 8833 motor driver, which allows for the control of the DC motors' turning velocity. Specifically, the speed of these motors is regulated by means of the two hardware PWM pins GPIO19 [PWM0] and GPIO12 [PWM1]. Thus, the larger the duty cycle of the hardware PWM signal, the faster the motors will spin. Then, the servomotor is directly connected to the RPi through the GPIO17 and controlled through a software PWM signal via the *pigpio* Python library.

7.2. Gondola 3D design

The design of the gondola frame and configuration followed an iterative process. First, an initial prototype was built by taking inspiration in two interesting works from the literature [66][49]. This first concept, nevertheless, turned out to be too heavy, large and non versatile. Therefore, this initial structure was iteratively optimized by minimizing its weight while keeping its sturdiness, and maximizing its versatility. The versatility property refers to the ease with which the gondola can be adapted to different applications by adding or modifying its components. These characteristics, as it will be seen in the upcoming section, make the proposed airship design ideal for robotics research.

7.2.1. Preliminary design: first prototype

Figure 7.5 presents a detailed 3D view of the first prototype's gondola frame, together with a top, side and bottom perspectives of the complete structure and its components. To obtain this concept, inspiration on the design was mainly taken from [49]. This configuration has a total span of 30cm and weight of 162g with all cables and electronics. Given that the maximum lift that the balloon can provide is of 150g, reducing some weight became fundamental, to obtain an equilibrium of forces in the vertical direction of the airship.

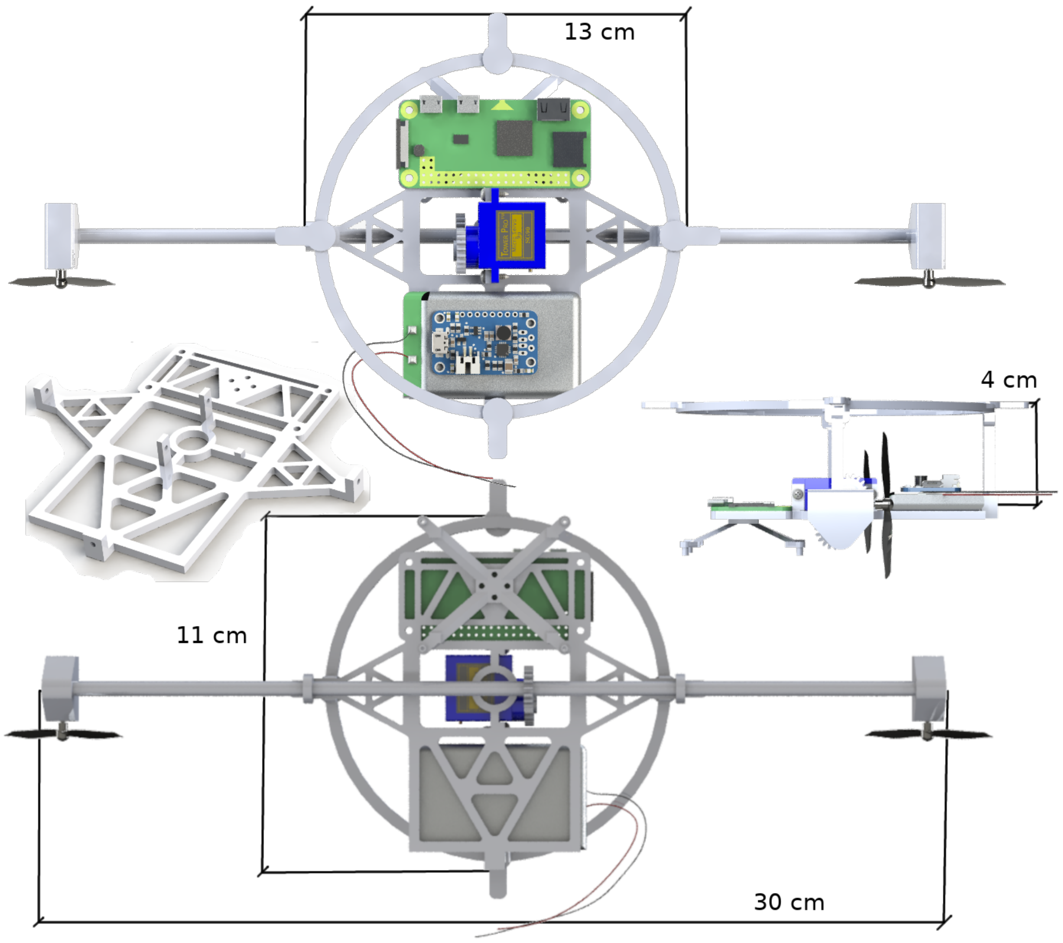


Figure 7.4: First prototype gondola configuration and different views.

7.2.2. Definitive gondola

Figure 7.5 shows the final configuration of the gondola, also via a 3D perspective of the frame, and the three main views of the global structure: top, side and bottom. The span of the new design is of 25cm and its total weight of 147g. Therefore, we managed to recover a total of 17g by optimizing the main frame, the top attachment and the diameter of the shaft.

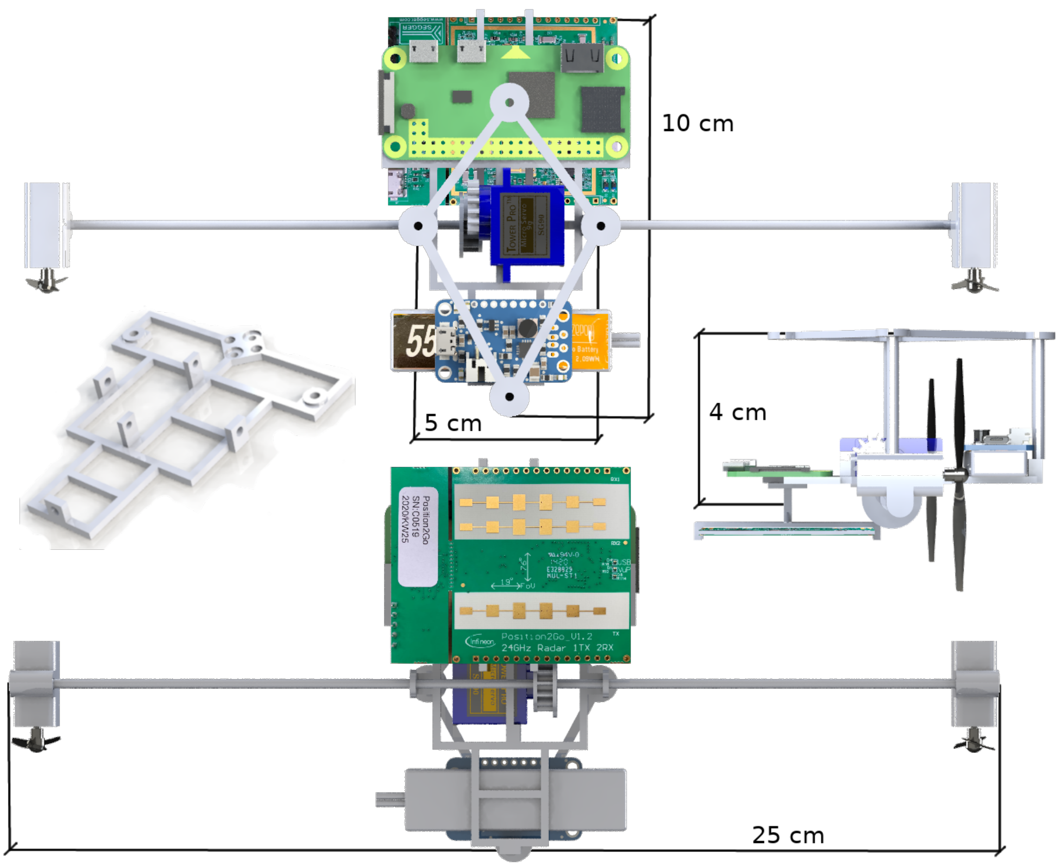


Figure 7.5: Final prototype gondola configuration and different views.

A clear contrast between the initial and final gondola frames is showcased in Figure 7.6. From this image, the diminution in both length and width becomes obvious but, more importantly, the notable reduction in the amount of material used (PLA).

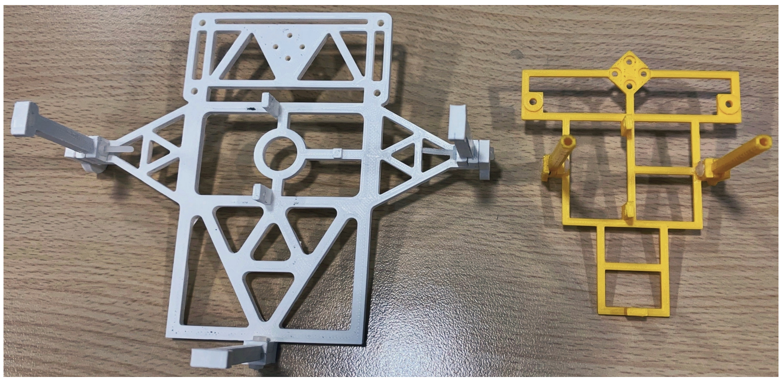
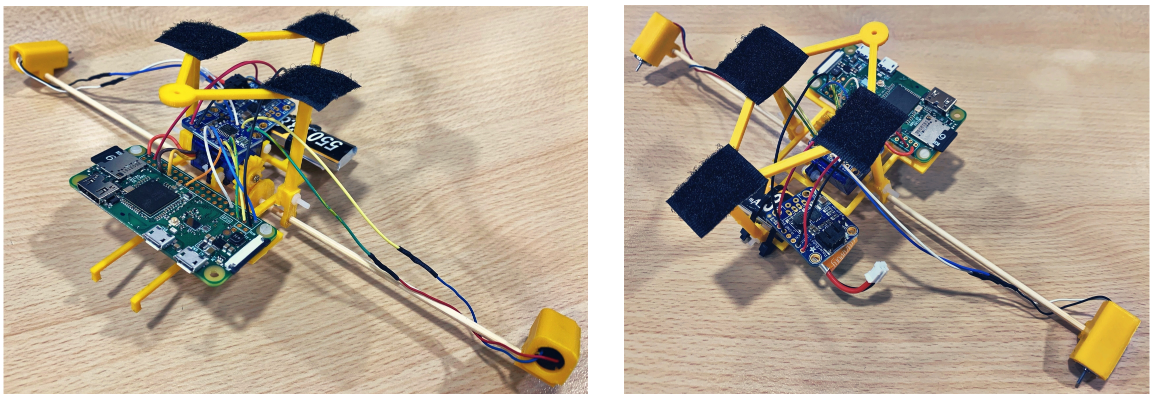


Figure 7.6: Size comparison between the first and definitive gondola frame designs.

From the definitive design, we see that the Raspberry Pi Zero W is placed at the front of the frame. Then comes the microservo which, as it was already explained at the beginning of the section, makes the shaft rotate (from 0°to 180°) thanks to a two-gear mechanism, which is also 3D printed. At the end of the rotary shaft, two cases hold the DC motors, to which the two propellers are attached. At the rear end, the step-up voltage regulator and the Li-Po battery that act as the power supply for the rest of components are located. The DRV8833 that controls the motors is not shown in the scheme. At the top, four hinges that allow to attach the gondola to the blimp's envelope by means for Velcro strips provide closure to the structure. Only the radar sensor is placed at the bottom, leaving enough space, so that the frame itself does not add noise to the altitude signal measurements. The radar, however, can also just be directly attached to the airship, also via a Velcro strip. As it can be observed, the different parts are mounted like a puzzle, in a convenient modular configuration.



(a) Front view.

(b) Rear view.

Figure 7.7: Definitive gondola mounted in reality.

Figure 7.7 displays the real gondola, once it is already mounted, with all electronic components and the cables soldered. Thus, we propose a light-weight and compact, modular, and simple gondola design, very convenient for the present application; but versatile enough so that it can easily be generalized for other projects with different sensors and/or actuators. Moreover, the proposed design is open-source `https://github.com/tudelft/blimp_snn`, and ideal for robotics research.

7.3. Airborne radar

7.3.1. Sensor specifications

The *Position2Go - XENSIV 24GHz* fast chirp FMCW radar module from *Infineon* for ranging, movement and target position estimation is used in this work. This platform has one transmitter (TX) and two receiver antennas (RX). Figure 7.8 illustrates the Position2Go platform and its main components. Table 7.3 summarizes some of the most important system specifications. Concretely, the board is capable of generating a 2D range-Doppler map to estimate the distance and velocity of targets. Moreover, the fact that it has two receiving antennas allows it to determine the angle of arrival (AoA) thanks to a phase monopulse comparison technique, as already explained in 3.2.3.

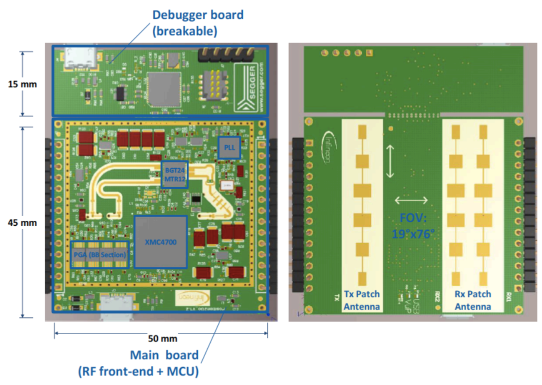


Figure 7.8: *Infineon Position2Go* board with main components and dimensions.[1]

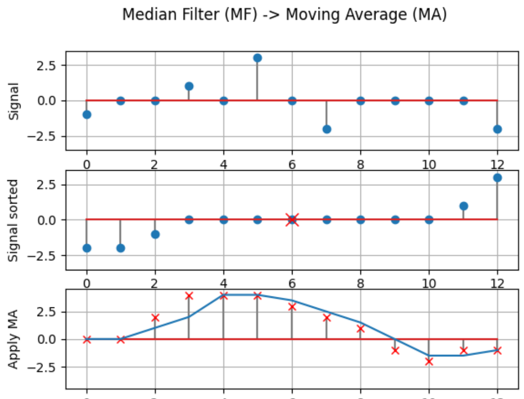
Speed $[km/h]$		Distance $[m]$		Range $[cm]$		Field of View $[°]$
Min.	Max.	Min.	Max.	Accuracy	Resolution	
0	36	0.6	15	± 20	90	19×76

Table 7.3: Position2Go module performance specifications.

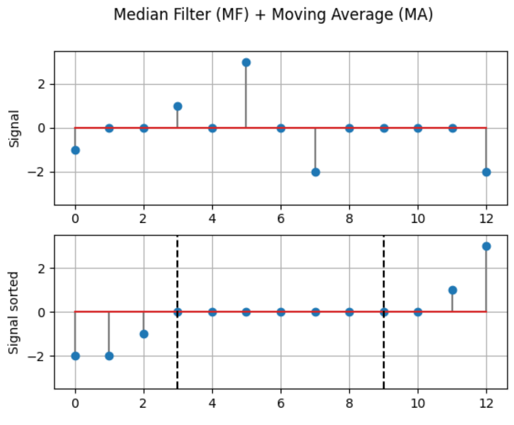
The values in Table 7.3 have been taken from the *Infineon Position2Go* specifications sheet.[2]

7.3.2. Signal processing and filtering

To process the measurements from the radar, a median filter (MF) is applied to remove possible outliers, and a moving average (MA) filter is used to smooth out the signal. The first option considered is that shown in Figure 7.9a where, first, a MF is applied over a first window -of size 13 points, in this example- . Basically, each of the measurement samples belonging to that window are first sorted out from the smallest to the largest obtained value, and that corresponding to the median is selected. Therefore, by choosing an appropriate size, the outliers will stay at the window extremes and not be considered. Then, a linear MA filter of size 2 is applied. From this figure, however, we can see that applying each of the filters sequentially does not seem to be the best option, as it increments the amount of delay present in the filtered signal.



(a) MF followed by MA filters, sequentially.



(b) MF and MA filters, applied simultaneously.

Figure 7.9: Exemplification of the radar's signal processing filters.

Therefore, to diminish the delay that is inherently generated when filtering the signal, a second and preferred option is to use the procedure showcased in Figure 7.9b. This is, to apply the MF and MA filters simultaneously. Therefore, in this example case, a first window of size 13 samples ($W_{MF} = 13$) is sorted and, from this set of points, a MA is performed over the points in the central part, such that the outliers are, again, not selected (the subwindow size is now 7, $W_{MA} = 7$). Figure 7.9 is just shown, however, as an example. The window size chosen for our particular case is: $W_{MF} = 15$ and $W_{MA} = 9$, after showing these values the best results.

Regarding the specific type of MA filter, two alternatives have been considered: applying a linear moving average (LMA) or an exponential moving average (EMA) filter. Equation 7.1 shows the mathematical expression for the LMA:

$$h_{filt}[n] = \frac{1}{n}\sum_{k=0}^{n} h[n-k], \tag{7.1}$$

with n, the total number of samples in the filtering window.

Then, the difference equation of an EMA filter corresponds to:

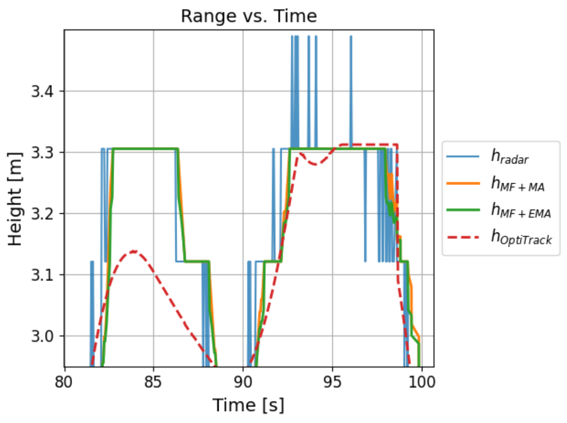
$$h_{filt}[n] = \alpha h[n] + (1-\alpha)h_{filt}[n-1] \tag{7.2}$$

[2] https://www.infineon.com/cms/en/product/evaluation-boards/demo-position2go/

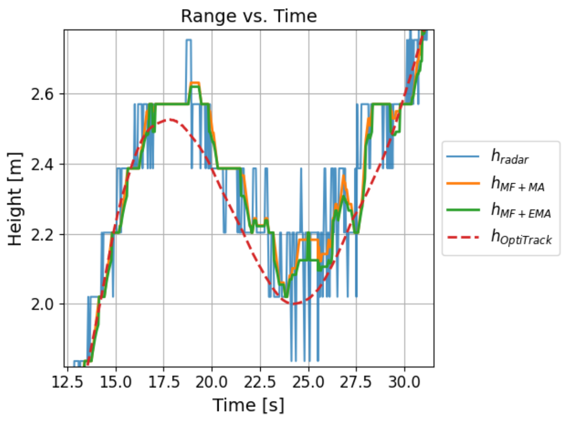
In Equation 7.2, $h_{filt}[n]$ is the current output, $h_{filt}[n-1]$ is the previous output, and $h[n]$ is the current input (the current non-filtered altitude measurement); α is a number between 0 and 1. If $\alpha = 1$, the output is just equal to the input, and no filtering takes place. Moreover, the filter is called *exponential*, because the weighting factor of previous inputs decreases exponentially. This can be easily demonstrated by substituting the previous outputs, as shown in Equation 7.3:

$$\begin{aligned} h_{filt}[n] &= \alpha h[n] + (1-\alpha)h_{filt}[n-1] \\ &= \alpha h[n] + (1-\alpha)(\alpha h[n-1] + (1-\alpha)h_{filt}[n-2]) \\ &= \alpha h[n] + (1-\alpha)(\alpha h[n-1] + (1-\alpha)(\alpha h[n-2] + (1-\alpha)h_{filt}[n-3])) \\ &= \alpha \sum_{k=0}^{n} (1-\alpha)^k h[n-k] \end{aligned} \tag{7.3}$$

Some results of the final filtered signal obtained after applying the MF together with the LMA and EMA filters are shown in Figure 7.10. In example #1 we observe how the MF takes care of removing the outliers that appear between the 90 and 100 seconds. From 82 to 88 there is a mismatch between the ground truth signal and that measured by the radar, but this is due to the resolution of the sensor, which is of 20 cm.



(a) Filtering example #1.



(b) Filtering example #2.

Figure 7.10: Results of the filtered signal for both the LMA and EMA cases on two real test flights.

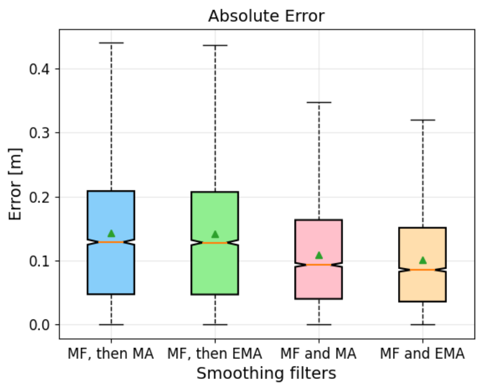


Figure 7.11: Boxplot of the error obtained with each of the considered filtering options with respect to the ground truth measured via OptiTrack.

To select the final filter used in this work, a boxplot of the absolute errors obtained for each of the alternatives is generated, as shown in Figure 7.11. Here we see that the simultaneous application of a MF and an EMA filters is best and, therefore, this is the final option selected.

8 Altitude controller

The training of the neural networks (both the spiking, SNN, and non-spipking, ANN) takes place in a simulated environment. Moreover, a model of the airship is generated to evaluate the performance of the controllers in simulation. The next two sections explain these parts of the project in further detail, to complement the discussions already provided in the scientific paper of Part I.

8.1. Simulation framework

As explained in Chapter 6, an evolutionary strategy is selected to train the neurocontrollers in simulation. This is done by means of the Distributed Evolutionary Algorithms in Python (DEAP) [56] framework. The PyTorch and PySNN [30] libraries are used to define the architectures of the ANN and SNN, respectively. Moreover, Algorithm 1 shows the step-by-step procedure that is followed to train the networks.

Algorithm 1 Evolutionary algorithm

```
Require: initialize(population)
Ensure: size(population) $==N$
  evaluate(offspring)
  for g in range(ngen) do
      offspring = select(population, size(population))
      offspring = mutate(offspring, $p^n_{mut}$, $p^m_{mut}$)
      evaluate(offspring)
      population = offspring
  end for
```

The first step is to randomly initialize a population of neurocontrollers of size N. Then, a first evaluation of this set of individuals is performed. Later, for each generation in a total set number of $ngen$ generations: 1) the offspring -with the same size as the population, N- is first generated following a randomized tournament selection (more details later, in Algorithm 2); 2) each of the offspring individuals mutates with a probability $p^{(n)}_{mut}$ and, each of the parameters of the mutated individuals, mutates with a probability $p^{(m)}_{mut}$; 3) a fitness is assigned to each offspring individual by evaluating its performance in a simulated altitude following task (see Algorithm 3); finally, this offspring becomes the new population.

Algorithm 2 Random selection tournament

```
  while i < N do
      aspirants = selectRandom(population, $M$)
      chosen ←append: max(aspirants, fitness)
      i+=1
  end while
```

The procedure for generating the new offspring each generation, by selecting individuals from the population is presented in Algorithm 2. Therefore, for a total amount of N times, which corresponds to the size of the population (this is, so that size(offspring)=size(population) at all times): 1) a total of M individuals are randomly selected from the population; 2) then, the best of this M individuals, or *aspirants*, is kept by choosing the one with the maximum fitness; 3) as said before, this is done for a total of N times. Finally, the evaluation procedure to compute the performance or *fitness* is showcased

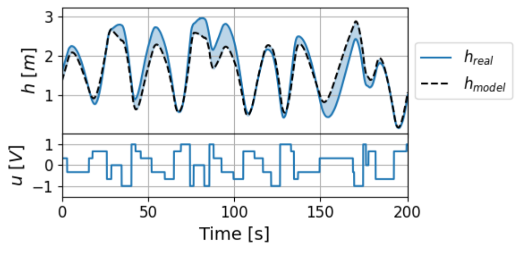
Algorithm 3 Fitness evaluation function

```
$h_{curr} = H$
for $h_{ref}$ in $h_{list}$ do
    while $t < T$ do
        $err = h_{ref} - h_{curr}$
        $u$ = controller($err$)
        $h_{curr}$ = model($u$)
        $h_{curr} +=$ random.uniform(noise)
        $err_{array} \leftarrow$ append: $err$
    end while
end for
RMSE = rmse_calculation($err_{array}$)
fitness = RMSE
```

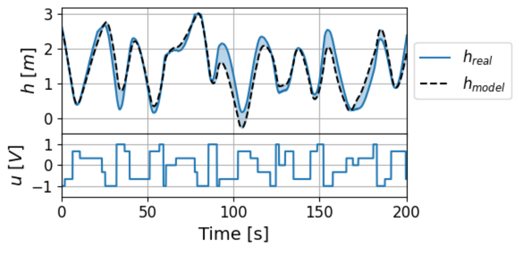
in Algorithm 3. The first step is to set an initial altitude, H, at which the simulation will start. Then, for every desired altitude h_{ref} in the total list of altitudes we want to evaluate and while the simulation time, t, is smaller than the maximum, T: 1) compute the error, which takes into account the sign, depending on whether we would like to go up, $err > 0$, or down, $err < 0$; 2) feed this error to the controller (ANN or SNN, in this case) to obtain the motor command, u; 3) compute the new altitude of the blimp, h_{curr} using the developed model of the plant (see Section 8.2 for more details); 4) add some noise to the altitude signal to increase the robustness; 5) save the current error in an error array and repeat for all timestamps and altitudes; 6) finally, compute the RMSE of this array of errors. The fitness corresponds to the RMSE.

8.2. Model of the plant

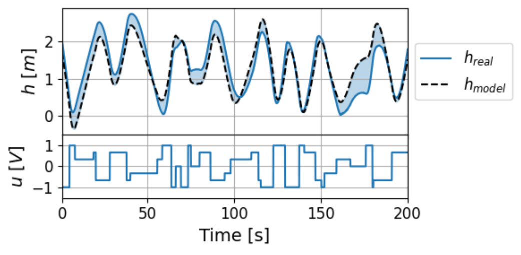
To train the neurocontrollers in simulation, a data-driven model of the blimp was generated.



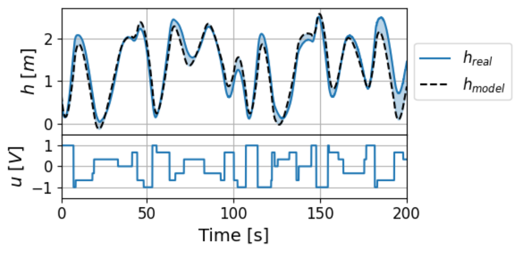
(a) Example run #1.



(b) Example run #2.



(c) Example run #3.



(d) Example run #4.

Figure 8.1: Contrast between the real and simulated altitude of the blimp, by taking as input identical motor commands, corresponding to the airship's manual teleoperation from a ground computer.

The scientific article in Part I already dives into some detailed explanations regarding the generation on the model. In this section, however, we provide some more visual examples on its performance. Figure 8.1 shows four test flight examples that contrast the ground truth altitude recorded in the Cyber-Zoo flight arena through manual teleoperation of the platform, and the simulated heights obtained via the developed model.

The procedure followed to determine the mathematical shape of the model's transfer function is showcased next. The objective is to obtain a mapping between the motor commands provided by the controller and the blimp's altitude over time. For that, we assume that the blimp's acceleration at the k-th time step, $\ddot{h}_k$, is proportional to the voltage applied to the motors, u, as:

$$\ddot{h}_k = \sum_i a_i u_{k-i} \tag{8.1}$$

where a_i is the proportionality constant for the motor command at time instant $k-i$. Nevertheless, the relation between $\ddot{h}$ and u cannot be directly inferred, given that the radar sensor cannot directly measure accelerations. By taking Euler's discretization of the derivative we can, however, express this relation in terms of the measured altitude and, finally, obtain the transfer function of the model by applying some simple mathematical operations, as seen in Equation 8.3.

$$\begin{aligned} \ddot{h}_k = \dot{h}_k - \dot{h}_{k-1} &= \sum_i a_i u_{k-i} \\ (h_k - h_{k-1}) - (h_{k-1} - h_{k-2}) &= \sum_i a_i u_{k-i} \\ h\left(1 - 2z^{-1} + z^{-2}\right) &= \sum_i a_i z^{-i} u \end{aligned} \tag{8.2}$$

$$h = \frac{\sum_i a_i z^{-i}}{1 - 2z^{-1} + z^{-2}} u, \tag{8.3}$$

The expression that is explicitly used in the *evaluation* function, during the simulated evolution, can be obtained after slightly manipulating the previous equations. Moreover, experimentally, we end up obtaining a model of order two. Following all this, we get:

$$(h_k - h_{k-1}) - (h_{k-1} - h_{k-2}) = a u_{k-1} + b u_{k-2} \rightarrow \tag{8.4}$$

$$h_k = 2h_{k-1} - h_{k-2} + a u_{k-1} + b u_{k-2} \tag{8.5}$$

$$h_k = h_{k-1} + \dot{h}_{k-1} + a u_{k-1} + b u_{k-2} \tag{8.6}$$

Therefore, Equation 8.6 is the one used to compute the new values of the current altitude, h_{curr} or h_k, and evaluate the performance of the individuals. This equation, requires some initial conditions (for $k = 0$), which are set to: $h_{-1} = H$, with H any initial desired altitude decide; $\dot{h}_{-1} = 0$, no initial velocity, $u_{-1} = u_{-2} = 0$, initial motor commands with a value of zero.

9 Discussion of experiments

The present chapter delineates the ROS architecture designed for the autonomous operation of the blimp with all of the proposed controllers, together with providing some further insight into the results presented in Part I, and the considered methodologies to ameliorate the reality gap.

9.1. Experimental setup with ROS

To perform the experiments, a suitable ROS architecture in which the all the different electronics -comprising the sensor, microcontroller, actuators, voltage regulator, driver, and battery- are appropriately integrated is proposed. A summary of this architecture explaining how the different parts are interconnected is showcased in the scheme of Figure 9.1.

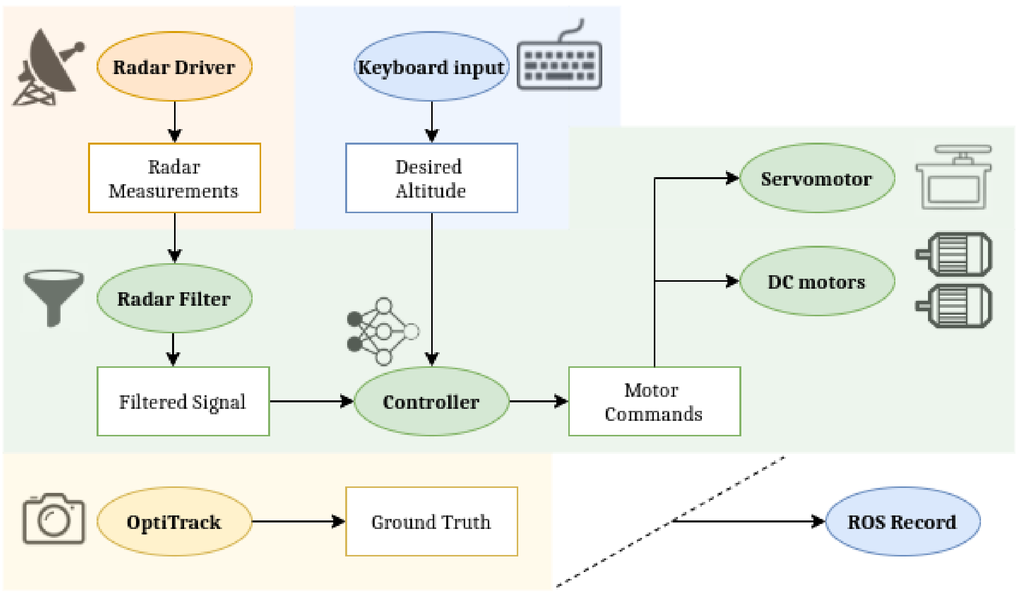


Figure 9.1: ROS architecture. Color orange represents the radar sensor; blue, the ground computer; green, the Raspberry Pi Zero W; yellow, the OptiTrack Motion Capture system.

First, the *Radar Driver* node, corresponds to the short-range 24 GHz FMCW radar sensor. From the Raspberry Pi Zero W, we launch this radar node, which publishes the raw and unfiltered range measurements in the *Radar Measurements* topic. Then, the median and moving average filters explained in Subsection 7.3.2 are launched within the *Radar Filter* node which, therefore, takes care of the altitude signal processing part. The filtered values of the altitude are then published into the *Filtered Signal* topic. Next, the controller -which, in this case, can correspond to a PID, ANN or SNN- gets the reference or desired altitude that we would like to achieve from the *Desired Altitude* topic, and the current height from the *Filtered Signal* topic. The desired altitude is set directly from a ground computer's keyboard, which is connected to the Raspberry Pi via a Secure Shell (SHH) connection. The *Keyboard Input* node takes care of this. Finally, the controller publishes the motor commands in the shape of voltages, into the *Motor Commands* topic, which is read by the *DC motors* and *Servomotor* nodes, that move the actuators. In parallel, the true range measurements from the OptiTrack Moption Capture system are published to the *Ground Truth* topic. As we can see, this part is not directly connected to any other node/topic in the main pipeline, since it is only used in the post-processing, for validation of the results. Lastly, all the messages or information published to the topics just discussed are recorded from a *Rosbag Record* node launched and saved directly to the ground computer. This is also used later for further analysis and/or post-processing.

9.2. Results and comparison

When transferring the controllers from the simulated environment to the actual airship, we will observe a mismatch commonly referred to as the [139][69][46]. How large or small this mismatch is depends mainly on how well we can replicate the real-world conditions in simulation. In our case, this gap causes the blimp to show an oscillatory behavior around the desired altitude. A good approach to ameliorate this undesired behavior, based on its success on previous works [139], would be that presented in Figure 9.2.

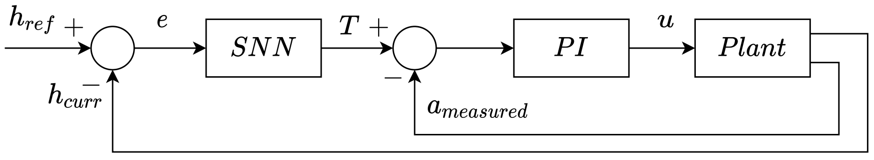


Figure 9.2: Strategy #1 for reducing the reality gap: sequential PI.

According to this scheme, the output of the neurocontroller would directly correspond to the thrust, T $[m/s^2]$, that must be applied to the blimp so that it follows the altitude setpoints. Then, with the acceleration measurements carried on-board the airship, a PI controller would eliminate the existing mismatch between the applied thrust and the actual detected acceleration, to finally send the appropriate motor command, u, to the robot. This solution, however, is not feasible in our work. The reason for this is that we are not carrying an accelerometer on-board and, therefore, the output of the neurocontroller cannot directly correspond to the thrust, T. Based on this, a second alternative is proposed in Figure 9.3.

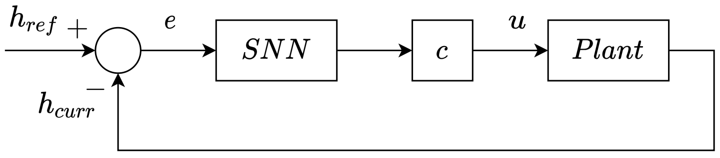


Figure 9.3: Strategy #2 for reducing the reality gap: scaling constant, c.

This case assumes that the oscillatory behavior of the blimp is caused by the motor commands given by the neurocontroller being either too large or too small. Therefore, a scaling constant $|c| < 1$ or $|c| > 1$, respectively, is proposed to counteract this effect. After testing it in reality, however, the reality gap increases. The reason for this is that u is not necessarily *always* too large or small, but it does not adapt perfectly well when the blimp gets very close to the desired height, because of certain phenomena occurring in reality that is not accounted for in simulation: the delay inherent to the actuators, the nonlinearities not accounted for in the linear model, the behavior of the radar signal not being perfectly modelled, the slight wind disturbances present in the flight arena, etc. Based on the insight gained by considering these two options, one final alternative to reduce the reality gap is proposed, as shown in Figure 9.4.

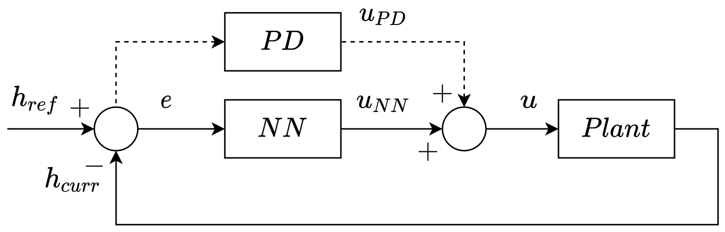


Figure 9.4: Strategy #3 for reducing the reality gap: parallel PD.

In this case, a PD controller directly tuned with in reality is added in parallel to the neurocontroller. The fact that its gains are tuned in reality, allows this block to improve the behavior of the airship when it approaches the altitude waypoints by accounting for the aforementioned reality phenomena that was not simulated when evolving the neurocontrollers. The PD gains, however, are chosen to be small such that its contribution to the total motor command can assure that the ANN/SNN is still the main contributor by far. In our case, the total PD contribution to the motor command, u, is of 16%. The difference between the results obtained without and with the parallel PID for the case of a SNN and an ANN controller, are showcased in Figure 9.5. This approach allows for an improvement that goes from a RMSE of 38 cm and 36 cm for the case of the SNN and ANN, respectively, to 27 cm for both cases. This is, even though the oscillations are still present, a total error reduction of a 29% for the SNN and 25% for the ANN is achieved.

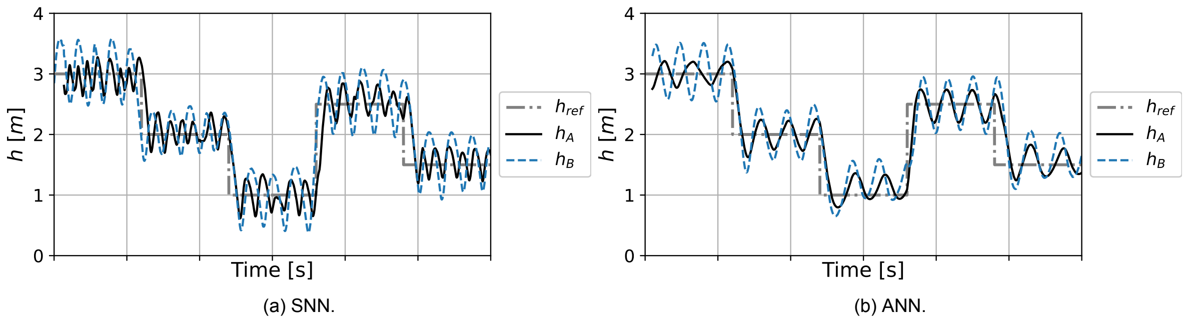


Figure 9.5: Contrast between the results with, h_A, and without, h_B, the reality gap reduction strategy.

Moreover, the results obtained for both the ANN and SNN for several runs on a different set of altitudes to those thoroughly analyzed in the scientific paper of Part I are presented in Figure 9.6, proving the robustness of the control method.

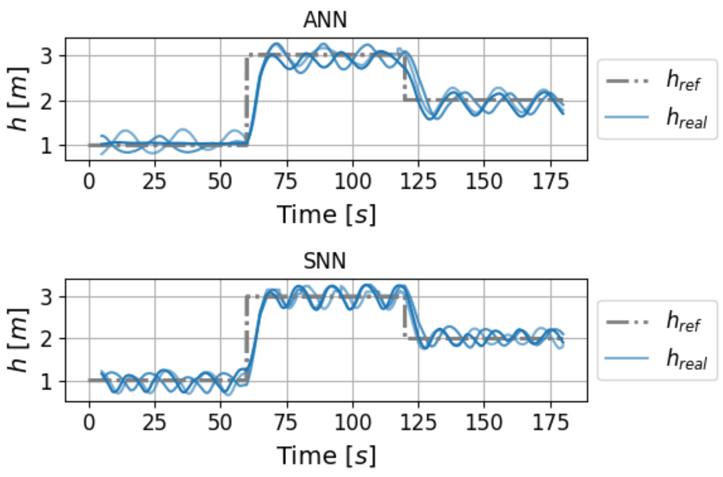
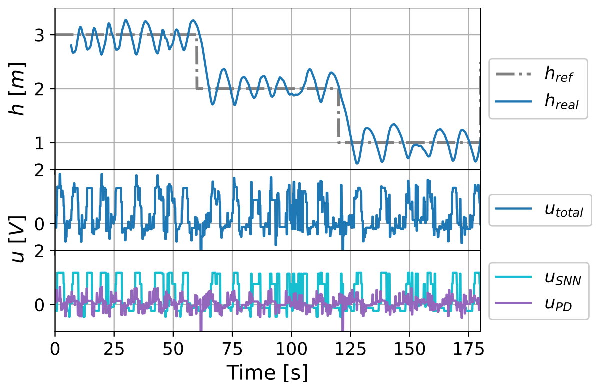
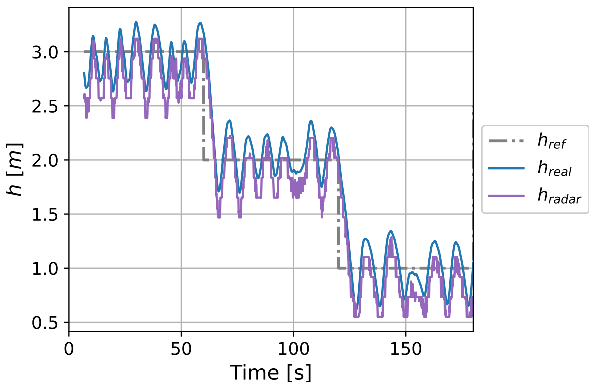


Figure 9.6: Results of the neurocontrollers on the altitude setpoints $[1, 3, 2]$ m.

An example of what the filtered radar signal, following the procedure delineated in Section 7.3.2 looks like is presented in Figure 9.7. Concretely, on the left we see the results for the altitude control with the SNN controller and its motor commands and, on the right, we show the contrast between the measured heights with the radar, on which the SNN relies, and the true altitude measured via the OptiTrack Motion Capture system. We can see how, after applying the median and exponential moving average filters, the measured signal accurately follows the ground truth altitude most of the time. Also, because of the radar resolution being of around ~ 20 cm, sometimes the measurements *jump* slightly more, showing some differences with OptiTrack; although there are no major outliers or remarkable contrasts between the two.



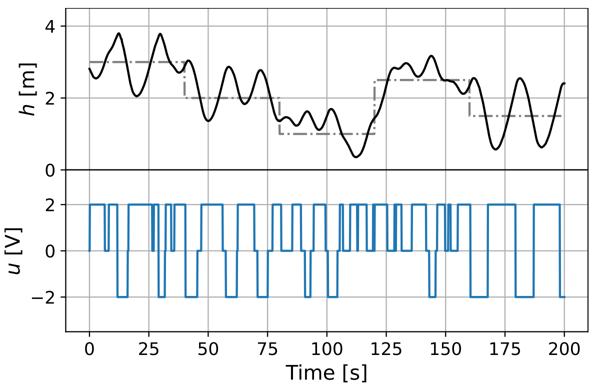
(a) SNN ground truth altitude and motor commands, u.



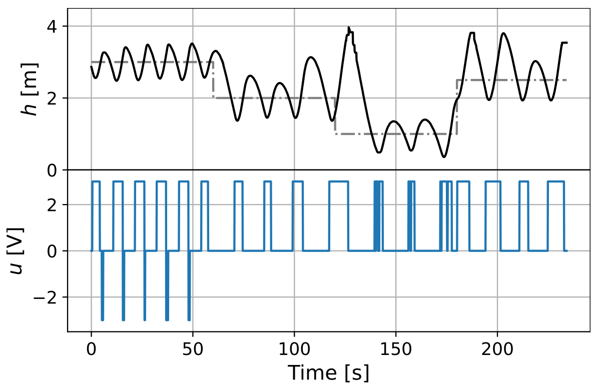
(b) Contrast between the ground truth and measured h.

Figure 9.7: SNN results for $h = [3, 2, 1]m$ zoomed in.

As explained in Part I, the decoding of the network is performed by means of a non-spiking neuron endowed with a *tanh* (hyperbolic tangent) activation function, whose weights are also evolved during simulation. Before this strategy would work, a different one based on a *population decoding* type of scheme was implemented, that would convert the spikes back to a real number, corresponding to the motor command, u



(a) Example run #1 with $u = [-2, 0, 2]V$.



(b) Example run #2 with $u = [-2.5, 0, 2.5]V$.

Figure 9.8: SNN control via a population decoding strategy.

Figure 9.8 show the experimental results for two SNN evolved in simulation with 3 output neurons. Each of these neurons outputs a discrete value of the motor command voltage, u, every time they fire. For example, on the left, when the first neuron fires it provides 2V to the DC motors and, when the middle or last one do, they output 0 and -2V, respectively.

From this image, we can see that the results obtained via the population decoding strategy are poor and do not allow for a reality gap reduction. This is what we would expect, since the mismatch between simulation and reality comes from other sources in the control loop (for example, the linear model, the delay inherent to the actuators, and more, as already explained before). Moreover, allowing only for a certain amount of discrete values of u, makes the control more inflexible and abrupt as we approach the reference altitudes. Some other population decoding schemes with more output neurons were tested allowing, thus, for additional discrete values. However, the performance of these controllers still stands far away to the one proposed in the scientific article (the ANN *tanh* neuron at the end) due to the non-continuous nature of the output. Moreover, the more neurons we add at the end, the more parameters will have to evolve and, therefore, the more complex the network will be.

Finally, the complete set of results for both simulation and reality, for the three considered types of controllers -SNN, ANN and PID- are showcased in Figure 9.9.

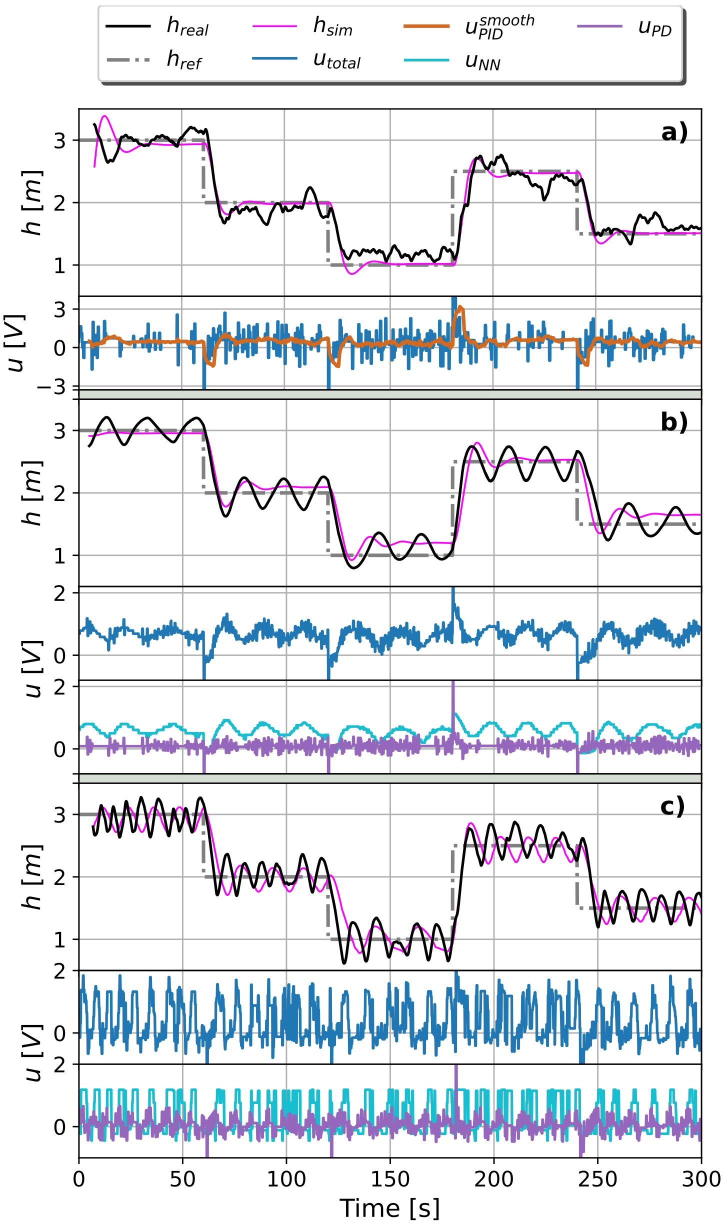


Figure 9.9: Experimental evaluation of the considered controllers. For all three subfigures, at the bottom we have the motor commands, and on top, the evolution of the blimp's ground altitude h_{real} and simulated altitude h_{sim}, compared with the reference h_{ref}. ***(a)*** u refers to the motor command, and u_{smooth} is obtained after smoothing it with a moving average. ***(b)*** u_{ANN} stands for the output of the evolved controller, u_{PD} to the contribution of the PD parallel controller, and $u_{total} = u_{ANN} + u_{PD}$. ***(c)*** Analogous to the ANN controller.

10 Conclusions

Despite the recent advancements, micro air vehicles (MAVs) still face the challenge of carrying on-board the complex controllers required to achieve autonomous flight operation. Spiking neural networks (SNNs) are arising as a promising paradigm, allowing for an efficient control while keeping the learning capabilities of classical networks. This work proposes an evolved altitude controller based on a SNN for a micro-robotic airship and its comparison with a non-spiking artificial neural network and a linear PID. The main contributions of this work are presented in the scientific article in Part I. This first part proves the advantages offered by the SNN controller in terms of the invested control effort. Then, in Part II we conduct a literature study of the relevant state-of-the-art, to provide the necessary theoretical background and contextualize our work. Finally, in Part III a detailed review of the methodology used to develop our work and accomplish the results presented is performed. More concretely, in this work we start by presenting an open-source design of a lightweight airship that easily customizes its on-board components. Then, we develop a SNN controller trained through an evolutionary strategy on a model-based simulation environment, which relies solely on the sensory feedback provided by an airborne radar sensor. Finally, we evaluate the system's performance in real-world experiments, comparing it with a standard PID and an artificial neural network. The results show that our novel SNN controller achieves a satisfactory performance with a low control effort. Our strategy can therefore be used to efficiently control the altitude of an airship equipped solely with a ranging sensor, and performing all the computations on-board.

10.1. Recommendations for future research

Following the methodology described in this report, the main thesis objective is successfully accomplished. However, the restricted timeline of the project, allows to tackle mainly the critical elements of the research objective. Therefore, certain parts offer room for further research that would allow for more accurate results and general improvements. This section provides some recommendations and guidelines for future research.

First and foremost, the area of the project that deserves more attention has to do with the reduction of the reality gap (RG), which appears when transferring the evolved controllers to the actual platform. This is because, as we can infer from the results, the behavior of the blimp continues to show some oscillations, even after applying the RG reduction procedure proposed in Chapter 9. Some recommendations for further research with the aim of palliating this RG are:

1. **Modify the selected RG reduction strategy: parallel PD**
   To alleviate the effects of the RG, the current strategy consists in adding a parallel PD controller to the ANN or SNN blocks such that the total motor command obtained corresponds to $u\ [V] = u_{NN} + u_{PD}$, as explained in Section 9.2. This solution, however, is not ideal as, preferably, the linear controller should be added sequentially, as also proposed in the aforenamed section. As already analyzed, we currently are unable to do this, as we do not carry a sensor that allows us to directly measure the acceleration on-board. Therefore, to implement this sequential control scheme, we propose:

- **Adding an Inertial Measurement Unit (IMU) sensor on-board**
  Currently, the only sensor is an on-board 24 GHz FMCW short-range radar for measuring the altitude. This implies that there is no way to measure the acceleration, other than differentiating twice the ground truth altitude received from the OptiTrack Motion Capture system. According to the thesis objective, however, all the relevant measurements and computations should be carried on-board the airship; so that this would not be a valid procedure. Moreover, conducting a double differentiation on the range measurement obtained from the radar would not provide valid acceleration measurements either, given that its resolution is of ~20 cm. This can easily be solved by adding an IMU sensor on-board to have accurate acceleration measurements available directly on the platform.
- **Directly getting the thrust, $u = T$, as the output from the ANN/SNN**
  Again, as we currently do not carry an IMU on-board, the motor command, u, obtained as the output from the neurocontrollers corresponds to the voltage that has to be applied to the motors such that they spin faster or slower. When adding the IMU, however, we would recommend to directly evolve the networks such that they provide the thrust, $T\ [N]$. Thanks to this, a PI controller could directly be added sequentially to ameliorate the RG, as shown in Section 9.2.

2. **Airship dynamics**

   Even though the implementation of the previous point would be the one which would make the largest difference in easing the small oscillations we still observe around the desired altitudes, a more realistic model of the airship would also contribute positively to our goal.

   - **Adding more complexity to the model**
     Currently, we are working with a linear model of the airship that has been developed by gathering data in the flight arena and finding the transfer function that best fits the data (see Part I for further details). Working with a linear model is already a simplification, as we are not accounting for any nonlinearities inherent to the robot. One option would therefore be to develop a nonlinear model of the airship.
   - **Working with an abstracted model**
     Another research direction that could be considered would be to directly develop an abstracted mathematical model, which is not data-driven. Accurately representing all the mass and inertias, gravity and buoyancy forces, aerodynamic damping and propulsion in this model would imply measuring a great number of coefficients, which could turn to be slightly impractical and require too much time (see Section 2.2). One could also use a simple abstracted model, but then we would go back to a similar situation to the one we currently have, which is a data-driven model of the airship, with some simplifications.

3. **Selection of the actuators and sensors**
   Building an airship with low cost components and materials (except for the sensor) easily available commercially, to facilitate the reproducibility of the platform was also among our objectives. This implies, however, that the quality of the actuators is not the highest. Moreover, they only spin in one direction. Therefore, selecting a different set of high-quality reversible DC motors could improve the delay inherent to the model and the control accuracy. A ranging sensor with a better resolution could be considered as well (the current one has ~20 cm).

4. **Simulated environment**
   Adding further randomization in simulation and/or further studying the noise/shape of the radar signal for a more accurate replication could also positively contribute to the RG reduction.

5. **Neuromorphic hardware**
   Currently, the Raspberry Pi Zero W on-board the airship is in charge of performing all the relevant calculations for the three controllers considered (PID, ANN and SNN). No specific hardware is included, however, to perform the computations of the SNN. The emergence of neuromorphic processors such as the Loihi [38] or the μBrain [150] yield new promises for the application of SNNs in robotics [46], by allowing to fully exploit their asynchronous properties and energy efficiency.

Therefore, an important step for future research would be to test our current implementation by also adding appropriate neuromorphic hardware in the loop.

IV

Appendices

Gantt chart

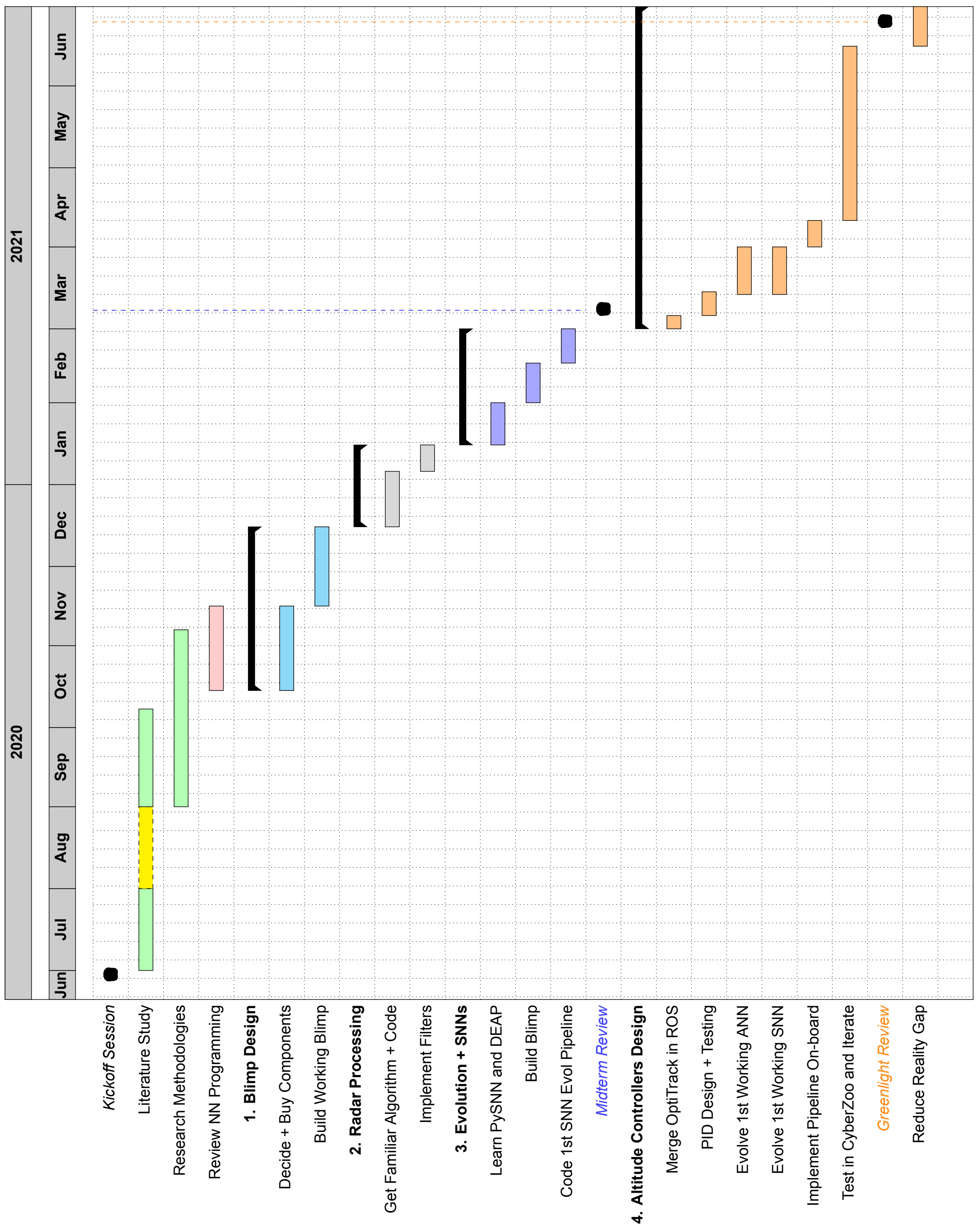


Figure 1: Gantt chart summarizing the structure, scope and timeline of the project.

Bibliography

[1] Air shark - the remote controlled fish blimp. https://faradayscienceshop.com/products/air-swimmer-the-remote-controlled-fish-blimp. Accessed: 2020-10-13.

[2] Blimp simulator ootang2018. https://github.com/ootang2018/blimp_simulator, . Accessed: 2020-10-13.

[3] Blimpduino 2.0 kit. https://www.jjrobots.com/blimpduino-2/, . Accessed: 2020-08-30.

[4] Position2go software user manual. https://www.infineon.com/dgdl/Infineon-P2G_Software_User_Manual-ApplicationNotes-v01_01-EN.pdf?fileId=5546d4626b2d8e69016b89493bf842af. Accessed: 2020-09-03.

[5] L. F. Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5-6):303—304, 1999. ISSN 0361-9230. doi: 10.1016/s0361-9230(99)00161-6. URL https://doi.org/10.1016/s0361-9230(99)00161-6.

[6] Fady Alnajjar and K. Murase. Self-organization of spiking neural network generating autonomous behavior in a miniature mobile robot. In Kazuyuki Murase, Kosuke Sekiyama, Tomohide Naniwa, Naoyuki Kubota, and Joaquin Sitte, editors, *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, pages 255–260, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-29344-6.

[7] Eleonora Arena, Paolo Arena, Roland Strauss, and Luca Patané. Motor-skill learning in an insect inspired neuro-computational control system. *Frontiers in Neurorobotics*, 11:12, 2017. ISSN 1662-5218. doi: 10.3389/fnbot.2017.00012. URL https://www.frontiersin.org/article/10.3389/fnbot.2017.00012.

[8] P. Arena, S. De Fiore, L. Patané, M. Pollino, and C. Ventura. Insect inspired unsupervised learning for tactic and phobic behavior enhancement in a hybrid robot. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010. ISBN 9781424469178. doi: 10.1109/IJCNN.2010.5596542.

[9] R. Batllori, C.B. Laramee, W. Land, and J.D. Schaffer. Evolving spiking neural networks for robot control. *Procedia Computer Science*, 6:329 – 334, 2011. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2011.08.060. URL http://www.sciencedirect.com/science/article/pii/S1877050911005254. Complex adaptive sysytems.

[10] Michel Baudry. Synaptic plasticity and learning and memory: 15 years of progress. *Neurobiology of Learning and Memory*, 70(1):113 – 118, 1998. ISSN 1074-7427. doi: https://doi.org/10.1006/nlme.1998.3842. URL http://www.sciencedirect.com/science/article/pii/S1074742798938424.

[11] C. Beck, H. J. Ng, R. Agethen, M. PourMousavi, H. P. Forstner, M. Wojnowski, K. Pressel, R. Weigel, A. Hagelauer, and D. Kissinger. Industrial mmwave radar sensor in embedded wafer-level bga packaging technology. *IEEE Sensors Journal*, 16(17):6566–6578, 2016. ISSN 1530437X. doi: 10.1109/JSEN.2016.2587731.

[12] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 153–160, Cambridge, MA, USA, 2006. MIT Press.

[13] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, 2014. ISSN 00189219. doi: 10.1109/JPROC.2014.2313565.

[14] S. Bermudez i Badia, P. Pyk, and P. F. M. J. Verschure. A biologically based flight control system for a blimp-based uav. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3053–3059, 2005. ISBN 078038914X. doi: 10.1109/ROBOT.2005.1570579.

[15] G. Q. Bi and M. M. Poo. Synaptic modification by correlated activity: Hebb's postulate revisited, 2001. ISSN 0147006X.

[16] Guo Qiang Bi and Mu Ming Poo. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, 1998. ISSN 02706474. doi: 10.1523/jneurosci.18-24-10464.1998.

[17] Z. Bing, C. Meschede, K. Huang, G. Chen, F. Rohrbein, M. Akl, and A. Knoll. End to end learning of spiking neural network based on r-stdp for a lane keeping vehicle. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4725–4732, 2018. ISBN 9781538630815. doi: 10.1109/ICRA.2018.8460482.

[18] Zhenshan Bing, Claus Meschede, Florian Röhrbein, Kai Huang, and Alois C. Knoll. A survey of robotics control based on learning-inspired spiking neural networks. *Frontiers in Neurorobotics*, 12:35, 2018. ISSN 1662-5218. doi: 10.3389/fnbot.2018.00035. URL https://www.frontiersin.org/article/10.3389/fnbot.2018.00035.

[19] Zhenshan Bing, Claus Meschede, Guang Chen, Alois Knoll, and Kai Huang. Indirect and direct training of spiking neural networks for end-to-end control of a lane-keeping vehicle. *Neural Networks*, 121:21–36, Jan 2020. ISSN 0893-6080. doi: 10.1016/j.neunet.2019.05.019. URL http://dx.doi.org/10.1016/j.neunet.2019.05.019.

[20] S. M. Bohte, J. A. L Poutre, and Joost N. Kok. Error-backpropagation in temporally encoded networks of spiking neurons. Technical report, NLD, 2000.

[21] Sander M. Bohte, Joost N. Kok, and Han La Poutre. SpikeProp: Backpropagation for networks of spiking neurons. *Neurocomputing*, 48:419–424, 01 2000.

[22] A. Bouganis and M. Shanahan. Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010. ISBN 9781424469178. doi: 10.1109/IJCNN.2010.5596525.

[23] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges. *ACM Journal on Emerging Technologies in Computing Systems*, 15(2):1–35, Jun 2019. ISSN 1550-4840. doi: 10.1145/3304103. URL http://dx.doi.org/10.1145/3304103.

[24] James M Bower and David Beeman. The Book of Genesis - Exploring Realistic Neural Models with the GEneral NEural SImulation System. *Genesis*, 2003. ISSN 00063495. doi: http://dx.doi.org/10.1016/S1364-6613(02)01915-0.

[25] Romain Brette, Michelle Rudolph, Ted Carnevale, Michael Hines, David Beeman, James M. Bower, Markus Diesmann, Abigail Morrison, Philip H. Goodman, Frederick C. Harris, Milind Zirpe, Thomas Natschläger, Dejan Pecevski, Bard Ermentrout, Mikael Djurfeldt, Anders Lansner, Olivier Rochel, Thierry Vieville, Eilif Muller, Andrew P. Davison, Sami El Boustani, and Alain Destexhe. Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 23(3):349–398, 2007. doi: 10.1007/s10827-007-0038-6. URL https://doi.org/10.1007/s10827-007-0038-6.

[26] Nicolas Brunel and Mark C. W. van Rossum. Lapicque's 1907 paper: From frogs to integrate-and-fire. *Biological Cybernetics*, 97(5):337–339, March 2008. ISSN 03401200. doi: 10.1007/s00422-007-0190-0.

[27] A. N. Burkitt. A review of the integrate-and-fire neuron model: Ii. inhomogeneous synaptic input and network properties. *Biological Cybernetics*, 95(2):97—112, August 2006. ISSN 0340-1200. doi: 10.1007/s00422-006-0082-8. URL https://doi.org/10.1007/s00422-006-0082-8.

[28] N. Burkitt. A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biol. Cybern.*, 95(1):1–19, June 2006. ISSN 0340-1200. doi: 10.1007/s00422-006-0068-6. URL https://doi-org.tudelft.idm.oclc.org/10.1007/s00422-006-0068-6.

[29] M. Burri, L. Gasser, M. Käch, M. Krebs, S. Laube, A. Ledergerber, D. Meier, R. Michaud, L. Mosimann, L. Müri, C. Ruch, A. Schaffner, N. Vuilliomenet, J. Weichart, K. Rudin, S. Leutenegger, J. Alonso-Mora, R. Siegwart, and P. Beardsley. Design and control of a spherical omnidirectional blimp. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1873–1879, 2013. ISBN 9781467363587.

[30] Bas Büller. Pysnn. https://github.com/BasBuller/PySNN, 2019.

[31] R.R. Carrillo, E. Ros, C. Boucheny, and O.J-M.D. Coenen. A real-time spiking cerebellum model for learning robot control. *Biosystems*, 94:18–27, 2008. ISSN 03032647. doi: 10.1016/j.biosystems.2008.05.008.

[32] George L. Chadderdon, Samuel A. Neymotin, Cliff C. Kerr, and William W. Lytton. Reinforcement learning of targeted movement in a spiking neuronal model of motor cortex. *PLOS ONE*, 7 (10):1–8, 10 2012. ISSN 19326203. doi: 10.1371/journal.pone.0047251. URL https://doi.org/10.1371/journal.pone.0047251.

[33] Sungjin Cho, Vivek Mishra, Qiuyang Tao, Paul Vamell, Matt King-Smith, Aneri Muni, Weston Smallwood, and Fumin Zhang. Autopilot design for A class of miniature autonomous blimps. In *1st Annual IEEE Conference on Control Technology and Applications, CCTA 2017*, 2017. ISBN 9781509021826. doi: 10.1109/CCTA.2017.8062564.

[34] Ting-Shuo Chou, Liam Bucci, and Jeffrey Krichmar. Learning touch preferences with a tactile robot using dopamine modulated stdp in a model of insular cortex. *Frontiers in Neurorobotics*, 9:6, 2015. ISSN 1662-5218. doi: 10.3389/fnbot.2015.00006. URL https://www.frontiersin.org/article/10.3389/fnbot.2015.00006.

[35] T. S. Clawson, S. Ferrari, S. B. Fuller, and R. J. Wood. Spiking neural network (snn) control of a flapping insect-scale robot. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3381–3388, 2016. ISBN 9781509018376. doi: 10.1109/CDC.2016.7798778.

[36] André Cyr, Mounir Boukadoum, and Frédéric Thériault. Operant conditioning: a minimal components requirement in artificial spiking neurons designed for bio-inspired robot's controller. *Frontiers in Neurorobotics*, 8:21, 2014. ISSN 1662-5218. doi: 10.3389/fnbot.2014.00021. URL https://www.frontiersin.org/article/10.3389/fnbot.2014.00021.

[37] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. ISSN 02721732. doi: 10.1109/MM.2018.112130359.

[38] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1): 82–99, 2018. doi: 10.1109/MM.2018.112130359.

[39] Van de Loo. Formation flight of two autonomous blimps. The Atalanta Wingman Project. *Technische Universiteit Eindhoven*, 2007.

[40] Arnaud Delorme and Simon J. Thorpe. SpikeNET: An event-driven simulation package for modelling large networks of spiking neurons. *Network: Computation in Neural Systems*, 2003. ISSN 0954898X. doi: 10.1088/0954-898X_14_4_301.

[41] Arnaud Delorme, Laurent Perrinet, and Simon J. Thorpe. Networks of integrate-and-fire neurons using rank order coding b: Spike timing dependent plasticity and emergence of orientation selectivity. *Neurocomputing*, 38-40:539 – 545, 2001. ISSN 0925-2312. doi: https://doi.org/10.1016/S0925-2312(01)00403-9. URL http://www.sciencedirect.com/science/article/pii/S0925231201004039. Computational Neuroscience: Trends in Research 2001.

[42] Howard B. Demuth, Mark H. Beale, Orlando De Jess, and Martin T. Hagan. *Neural Network Design*. Martin Hagan, Stillwater, OK, USA, 2nd edition, 2014. ISBN 0971732116.

[43] Di Hu, Xu Zhang, Ziye Xu, S. Ferrari, and P. Mazumder. Digital implementation of a spiking neural network (snn) capable of spike-timing-dependent plasticity (stdp) learning. In *14th IEEE International Conference on Nanotechnology*, pages 873–876, 2014. ISBN 9781479956227. doi: 10.1109/NANO.2014.6968000.

[44] Peter Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9:99, 2015. ISSN 1662-5188. doi: 10.3389/fncom.2015.00099. URL https://www.frontiersin.org/article/10.3389/fncom.2015.00099.

[45] Hebb D.O. *The Organization of Behavior*. 1949.

[46] Julien Dupeyroux, Jesse J. Hagenaars, Federico Paredes-Vallés, and Guido de Croon. Neuromorphic control for optic-flow-based landings of mavs using the loihi processor. *CoRR*, abs/2011.00534, 2020.

[47] Richard Evans. Reinforcement learning in a neurally controlled robot using dopamine modulated stdp, 2015.

[48] Faramarz Faghihi, Ahmed A. Moustafa, Ralf Heinrich, and Florentin Wrgtter. A computational model of conditioning inspired by drosophila olfactory system. *Neural Networks*, 87 (C):96–108, March 2017. ISSN 0893-6080. doi: 10.1016/j.neunet.2016.11.002. URL https://doi-org.tudelft.idm.oclc.org/10.1016/j.neunet.2016.11.002.

[49] Safwan Ul Ferdous, Alireza Mohammadi, and Sridhar Lakshmanan. Developing a low-cost autonomous blimp with a reduced number of actuators. In Charles M. Shoemaker, Hoa G. Nguyen, and Paul L. Muench, editors, *Unmanned Systems Technology XXI*, volume 11021, pages 73 – 80. International Society for Optics and Photonics, SPIE, 2019. ISBN 9781510627079.

[50] David Ferster and Nelson Spruston. Cracking the neuronal code. *Science*, 270(5237):756–757, 1995. ISSN 0036-8075. doi: 10.1126/science.270.5237.756. URL https://science.sciencemag.org/content/270/5237/756.

[51] Dario Floreano and Claudio Mattiussi. Evolution of spiking neural controllers for autonomous vision-based robots. In Takashi Gomi, editor, *Evolutionary Robotics. From Intelligent Robotics to Artificial Life*, pages 38–61, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-45502-8.

[52] Dario Floreano, Yann Epars, Jean-Christophe Zufferey, and Claudio Mattiussi. Evolution of spiking neural circuits in autonomous mobile robots: Research articles. *Int. J. Intell. Syst.*, 21(9): 1005–1024, September 2006. ISSN 0884-8173. doi: 10.1002/int.20173.

[53] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, Mar 2008.

[54] R. V. Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007. ISSN 1530888X. doi: 10.1162/neco.2007.19.6.1468.

[55] G. Foderaro, C. Henriquez, and S. Ferrari. Indirect training of a spiking neural network for flight control via spike-timing-dependent synaptic plasticity. In *49th IEEE Conference on Decision and Control (CDC)*, pages 911–917, 2010. ISBN 9781424477456.

[56] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. Deap: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13(70):2171–2175, 2012.

[57] Fossen. *Guidance and control of ocean vehicles*. 1995.

[58] Paula Fraga-Lamas, Lucía Ramos, Víctor M. Mondéjar-Guerra, and Tiago Fernández-Caramés. A review on iot deep learning uav systems for autonomous obstacle detection and collision avoidance. *Remote Sensing*, 11:21–44, 09 2019. doi: 10.3390/rs11182144.

[59] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, May 2014. ISSN 1558-2256. doi: 10.1109/JPROC.2014.2304638.

[60] Wulfram Gerstner and Werner M. Kistler. Mathematical formulations of Hebbian learning. *Biological Cybernetics*, 2002. ISSN 03401200. doi: 10.1007/s00422-002-0353-y.

[61] Kistler W. M. Gerstner W. *Spiking neuron models: Single neurons, populations, plasticity*. 2002. doi: 10.1017/9780511815706.

[62] Marc-Oliver Gewaltig and Markus Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007. ISSN 1941-6016. doi: 10.4249/scholarpedia.1430.

[63] P. González, W. Burgard, R. Sanz, and J. L. Fernández. Developing a low-cost autonomous indoor blimp. In *Journal of Physical Agents*, 2009. doi: 10.14198/JoPha.2009.3.1.06.

[64] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618.

[65] Dan Goodman and Romain Brette. Brian: a simulator for spiking neural networks in python. *Frontiers in Neuroinformatics*, 2:5, 2008. ISSN 1662-5196. doi: 10.3389/neuro.11.005.2008. URL https://www.frontiersin.org/article/10.3389/neuro.11.005.2008.

[66] G. Gorjup and M. Liarokapis. A low-cost, open-source, robotic airship for education and research. *IEEE Access*, 8:70713–70721, 2020. ISSN 21693536.

[67] Weibin Gu, Kimon P. Valavanis, Matthew J. Rutherford, and Alessandro Rizzo. Uav model-based flight control with artificial neural networks: A survey. *Journal of Intelligent & Robotic Systems*, 2020. doi: 10.1007/s10846-020-01227-8. URL https://doi.org/10.1007/s10846-020-01227-8.

[68] Amelie Hagelauer, Robert Weigel, Maciej Wojnowski, Klaus Pressel, and Dietmar Kissinger. Integrated systems-in-package. *IEEE Microwave Magazine*, 2018. ISSN 15579581. doi: 10.1109/MMM.2017.2759558.

[69] Jesse J. Hagenaars, Federico Paredes-Valles, Sander M. Bohte, and Guido C. H. E. de Croon. Evolved neuromorphic control for high speed divergence-based landings of mavs. *IEEE Robotics and Automation Letters*, 5(4):6239–6246, Oct 2020. ISSN 2377-3774.

[70] J. Hasch, E. Topak, R. Schnabel, T. Zwick, R. Weigel, and C. Waldschmidt. Millimeter-wave technology for automotive radar sensors in the 77 ghz frequency band. *IEEE Transactions on Microwave Theory and Techniques*, 60(3):845–860, 2012. ISSN 00189480. doi: 10.1109/TMTT.2011.2178427.

[71] Hananel Hazan, Daniel J. Saunders, Hassaan Khan, Devdhar Patel, Darpan T. Sanghavi, Hava T. Siegelmann, and Robert Kozma. Bindsnet: A machine learning-oriented spiking neural networks library in python. *Frontiers in Neuroinformatics*, 12:89, 2018. ISSN 1662-5196. doi: 10.3389/fninf.2018.00089. URL https://www.frontiersin.org/article/10.3389/fninf.2018.00089.

[72] S. Hazra and A. Santra. Short-range radar-based gesture recognition system using 3d cnn with triplet loss. *IEEE Access*, 7:125623–125633, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2938725.

[73] Michael Hines and Ted Carnevale. *NEURON Simulation Environment*, pages 1–8. Springer New York, New York, NY, 2013. ISBN 978-1-4614-7320-6. doi: 10.1007/978-1-4614-7320-6_795-1. URL https://doi.org/10.1007/978-1-4614-7320-6_795-1.

[74] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002. ISSN 0899-7667. doi: 10.1162/089976602760128018. URL https://doi-org.tudelft.idm.oclc.org/10.1162/089976602760128018.

[75] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL https://doi-org.tudelft.idm.oclc.org/10.1162/neco.2006.18.7.1527.

[76] Martin Hitzler, Linus Boehm, Winfried Mayer, and Christian Waldschmidt. Radiation Pattern Optimization for QFN Packages with On-Chip Antennas at 160 GHz. *IEEE Transactions on Antennas and Propagation*, PP:1–1, 06 2018. ISSN 0018926X. doi: 10.1109/TAP.2018.2846812.

[77] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952. doi: 10.1113/jphysiol.1952.sp004764. URL https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1952.sp004764.

[78] David Howard and Alberto Elfes. Evolving spiking networks for turbulence-tolerant quadrotor control. *Artificial Life Conference Proceedings*, (26):431–438, 2014.

[79] David Howard and Farid Kendoul. Towards evolved time to contact neurocontrollers for quadcopters. In Tapabrata Ray, Ruhul Sarker, and Xiaodong Li, editors, *Artificial Life and Computational Intelligence*, pages 336–347, Cham, 2016. Springer International Publishing. ISBN 978-3-319-28270-1.

[80] Anping Huang, Xinjiang Zhang, Runmiao Li, and Yu Chi. *Memristor Neural Network Design*. 04 2018. ISBN 978-953-51-3947-8. doi: 10.5772/intechopen.69929.

[81] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, Oct 2017. ISSN 2377-3774. doi: 10.1109/lra.2017.2720851. URL http://dx.doi.org/10.1109/LRA.2017.2720851.

[82] P. Hügler, M. Geiger, and C. Waldschmidt. 77 ghz radar-based altimeter for unmanned aerial vehicles. In *2018 IEEE Radio and Wireless Symposium (RWS)*, pages 129–132, 2018. ISBN 9781538607091. doi: 10.1109/RWS.2018.8304965.

[83] P. Hügler, F. Roos, M. Schartel, M. Geiger, and C. Waldschmidt. Radar taking off: New capabilities for uavs. *IEEE Microwave Magazine*, 19(7):43–53, 2018. doi: 10.1109/MMM.2018.2862558.

[84] Imec. Imec builds world's first spiking neural network-based chip for radar signal processing. URL https://www.imec-int.com/en/articles/imec-builds-world-s-first-spiking-neural-network-based-chip-for-radar-signal-pr

[85] E. M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14 (6):1569–1572, 2003. ISSN 10459227. doi: 10.1109/TNN.2003.820440.

[86] E.M. Izhikevich. *Dynamical Systems in Neuroscience*. Computational neuroscience Dynamical systems in neuroscience. MIT Press, 2007. ISBN 9780262090438. URL https://books.google.nl/books?id=kVjM6DFk-twC.

[87] Eugene M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 2004. ISSN 10459227. doi: 10.1109/TNN.2004.832719.

[88] Eugene M. Izhikevich and Gerald M. Edelman. Large-scale model of mammalian thalamocortical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 2008. ISSN 00278424. doi: 10.1073/pnas.0712231105.

[89] Jianguo Xin and M. J. Embrechts. Supervised learning with spiking neural networks. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, volume 3, pages 1772–1777, 2001. doi: 10.1109/ijcnn.2001.938430.

[90] Sophie Jordan, Julian Moore, Sierra Hovet, John Box, Jason Perry, Kevin Kirsche, Dexter Lewis, and Zion Tsz Ho Tse. State-of-the-art technologies for UAV inspections, 2018. ISSN 17518784.

[91] Andrzej Kasinski and Filip Ponulak. Experimental demonstration of learning properties of a new supervised learning method for the spiking neural networks. In Włodzisław Duch, Janusz Kacprzyk, Erkki Oja, and Sławomir Zadrożny, editors, *Artificial Neural Networks: Biological Inspirations – ICANN 2005*, pages 145–152, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-28754-4.

[92] Richard Kempter, Wulfram Gerstner, and J. Leo van Hemmen. Hebbian learning and spiking neurons. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 1999. ISSN 1063651X. doi: 10.1103/PhysRevE.59.4498.

[93] Dermot Kerr, M. McGinnity, Sonya Coleman, Qingxiang Wu, and M. Clogenson. Spiking hierarchical neural network for corner detection. *NCTA 2011 - Proceedings of the International Conference on Neural Computation Theory and Applications*, pages 230–235, 01 2011.

[94] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J. Thorpe, and Timothée Masquelier. Stdp-based spiking deep neural networks for object recognition. *CoRR*, abs/1611.01421, 2016. ISSN 18792782. doi: 10.1016/j.neunet.2017.12.005. URL http://arxiv.org/abs/1611.01421.

[95] Werner M. Kistler. Spike-timing dependent synaptic plasticity: A phenomenological framework. *Biological Cybernetics*, 2002. ISSN 03401200. doi: 10.1007/s00422-002-0359-5.

[96] Bruce W. Knight. Dynamics of encoding in a population of neurons. *The Journal of General Physiology*, 59(6):734—766, June 1972. ISSN 0022-1295. doi: 10.1085/jgp.59.6.734.

[97] J. Ko, D. J. Klein, D. Fox, and D. Haehnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 742–747, 2007. ISBN 1424406021. doi: 10.1109/ROBOT.2007.363075.

[98] Stuart Krause, Fabian Hartmann, and Jan-Peter Mund. UAV Workflow Optimization for the Acquisition of High-Quality Photogrammetric Point Clouds in Forestry. *GI_Forum*, 2016. ISSN 2308-1708. doi: 10.1553/giscience2016_01_s72.

[99] J. Kruger and F. Aiple. Multimicroelectrode investigation of monkey striate cortex: spike train correlations in the infragranular layers. *Journal of Neurophysiology*, 60(2):798–828, 1988. doi: 10.1152/jn.1988.60.2.798. PMID: 3171651.

[100] L. Lapicque. Recherches quantitatives sur l'excitation electrique des nerfs traitée comme une polarization. *Journal de Physiologie et Pathologie General*, 9:620–635, 1907.

[101] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10:508, 2016. ISSN 1662-453X.

[102] Yuwen Li, Meyer Nahon, and Inna Sharf. Airship dynamics modeling: A literature review. *Progress in Aerospace Sciences*, 47(3):217 – 239, 2011. ISSN 03760421.

[103] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008. ISSN 00189200. doi: 10.1109/JSSC.2007.914337.

[104] W. Liu, C. Yu, X. Wang, Y. Zhang, and Y. Yu. The altitude hold algorithm of uav based on millimeter wave radar sensors. In *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 1, pages 436–439, 2017. ISBN 9781538630228. doi: 10.1109/IHMSC.2017.106.

[105] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234, 12 2016. doi: 10.1016/j.neucom.2016.12.038.

[106] S. Loiselle, J. Rouat, D. Pressnitzer, and S. Thorpe. Exploration of rank order coding with spiking neural networks for speech recognition. 2006. doi: 10.1109/ijcnn.2005.1556220.

[107] Wofgang Maas. Networks of spiking neurons: The third generation of neural network models. *Trans. Soc. Comput. Simul. Int.*, 14(4):1659–1671, December 1997. ISSN 0740-6797.

[108] Wolfgang Maass and Christopher M. Bishop. *Pulsed Neural Networks*. MIT Press, 1999. ISBN 0262133504. doi: 10.7551/mitpress/5704.001.0001.

[109] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.*, 14 (11):2531–2560, November 2002. ISSN 0899-7667. doi: 10.1162/089976602760407955. URL https://doi-org.tudelft.idm.oclc.org/10.1162/089976602760407955.

[110] Urszula Markowska-Kaczmar and Mateusz Koldowski. Spiking neural network vs multilayer perceptron: who is the winner in the racing car computer game. *Soft Computing*, 19(12): 3465–3478, 2015. doi: 10.1007/s00500-014-1515-2. URL https://doi.org/10.1007/s00500-014-1515-2.

[111] Henry Markram, Joachim Lübke, Michael Frotscher, and Bert Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, 275(5297):213–215, 1997. ISSN 0036-8075. doi: 10.1126/science.275.5297.213. URL https://science.sciencemag.org/content/275/5297/213.

[112] Timothée Masquelier and Simon J Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLOS Computational Biology*, 3(2):1–11, 02 2007. doi: 10.1371/journal.pcbi.0030031. URL https://doi.org/10.1371/journal.pcbi.0030031.

[113] P. Mazumder, D. Hu, I. Ebong, X. Zhang, Z. Xu, and S. Ferrari. Digital implementation of a virtual insect trained by spike-timing dependent plasticity. *Integration*, 54:109 – 117, 2016. ISSN 0167-9260. doi: https://doi.org/10.1016/j.vlsi.2016.01.002. URL http://www.sciencedirect.com/science/article/pii/S0167926016000043.

[114] B. Meftah, O. Lezoray, and A. Benyettou. Segmentation and edge detection based on spiking neural network model. *Neural Processing Letters*, 32(2):131–146, 2010. doi: 10.1007/s11063-010-9149-6. URL https://doi.org/10.1007/s11063-010-9149-6.

[115] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated

circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014. ISSN 0036-8075. doi: 10.1126/science.1254642. URL https://science.sciencemag.org/content/345/6197/668.

[116] Vladimir Milovanovic. On fundamental operating principles and range-doppler estimation in monolithic frequency-modulated continuous-wave radar sensors. *Facta universitatis - series: Electronics and Energetics*, 31:547–570, 01 2018. ISSN 0353-3670. doi: 10.2298/fuee1804547m.

[117] Anthony Mouraud, Didier Puzenat, and Hélène Paugam-Moisy. DAMNED: A distributed and multithreaded neural event-driven simulation framework. *CoRR*, abs/cs/0512018, 2005. URL http://arxiv.org/abs/cs/0512018.

[118] Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, and Timothée Masquelier. Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Frontiers in Neuroscience*, 13:625, 2019. ISSN 1662-453X. doi: 10.3389/fnins.2019.00625. URL https://www.frontiersin.org/article/10.3389/fnins.2019.00625.

[119] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

[120] Floreano Nolfi. Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines. *MIT Press*, 2000.

[121] S. Oh, S. Kang, K. Lee, S. Ahn, and E. Kim. Flying display: Autonomous blimp with real-time visual tracking and image projection. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 131–136, 2006. ISBN 142440259X.

[122] Federico Paredes-Valles, Kirk Yannick Willehm Scheper, and Guido C.H.E. De Croon. Neuromorphic Computing of Event-Based Data for High-Speed Vision-Based Navigation. *Delft University of Technology (TU Delft)*, 2018.

[123] Federico Paredes-Valles, Kirk Yannick Willehm Scheper, and Guido C.H.E. De Croon. Unsupervised Learning of a Hierarchical Spiking Neural Network for Optical Flow Estimation: From Events to Global Motion Perception. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. ISSN 19393539. doi: 10.1109/TPAMI.2019.2903179.

[124] Prachi Patel. Agriculture drones are finally cleared for takeoff [News], 2016. ISSN 00189235.

[125] L. Perrinet, A. Delorme, M. Samuelides, and S.J. Thorpe. Networks of integrate-and-fire neuron using rank order coding a: How to implement spike time dependent hebbian plasticity. *Neurocomputing*, 38-40:817 – 822, 2001. ISSN 0925-2312. doi: https://doi.org/10.1016/S0925-2312(01)00460-X. URL http://www.sciencedirect.com/science/article/pii/S092523120100460X. Computational Neuroscience: Trends in Research 2001.

[126] Cyrill Planta, Jörg Conradt, Adrian Jencik, and Paul Verschure. A neural model of the fly visual system applied to navigational tasks. In José R. Dorronsoro, editor, *Artificial Neural Networks — ICANN 2002*, pages 1268–1274, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-46084-8. doi: 10.1007/3-540-46084-5_205.

[127] Filip Ponulak. ReSuMe-new supervised learning method for Spiking Neural Networks. *Inst. Control Information Engineering, Poznan Univ.*, 2005. ISSN 1530-888X. doi: 10.1.1.60.6325.

[128] Filip Ponulak and Andrzej Kasiński. Generalization properties of spiking neurons trained with resume method. In *ESANN 2006 Proceedings - European Symposium on Artificial Neural Networks*, 2006. ISBN 2930307064.

[129] Dimitri Probst, Wolfgang Maass, Henry Markram, and Marc-Oliver Gewaltig. Liquid computing in a simplified model of cortical layer iv: Learning to balance a ball. In Alessandro E. P. Villa, Włodzisław Duch, Péter Érdi, Francesco Masulli, and Günther Palm, editors, *Artificial Neural Networks and Machine Learning – ICANN 2012*, pages 209–216, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33269-2. doi: 10.1007/978-3-642-33269-2_27.

[130] Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, 9:141, 2015. ISSN 1662-453X. doi: 10.3389/fnins.2015.00141. URL https://www.frontiersin.org/article/10.3389/fnins.2015.00141.

[131] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[132] Axel Rottmann, Matthias Sippel, Thorsten Zitterell, and Wolfram Burgard. Towards an Experimental Autonomous Blimp Platform. *Emcr*, 2007.

[133] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Representations by Back-Propagating Errors*, page 696–699. MIT Press, Cambridge, MA, USA, 1988. ISBN 0262010976.

[134] Svein I. Sagatun and Thor I. Fossen. Lagrangian formulation of underwater vehicles' dynamics. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 1991. doi: 10.1109/icsmc.1991.169823.

[135] H. Saiki, T. Fukao, T. Urakubo, and T. Kohno. A path following control method under wind disturbances for outdoor blimp robots. In *2011 IEEE/SICE International Symposium on System Integration (SII)*, pages 978–984, 2011. ISBN 9781457715235. doi: 10.1109/SII.2011.6147582.

[136] M. Sarim, T. Schultz, R. Jha, and M. Kumar. Ultra-low energy neuromorphic device based navigation approach for biomimetic robots. In *2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*, pages 241–247, 2016. ISBN 9781509034413. doi: 10.1109/NAECON.2016.7856805.

[137] A. F. Scannapieco, A. Renga, G. Fasano, and A. Moccia. Ultralight radar sensor for autonomous operations by micro-uas. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 727–735, 2016. ISBN 9781467393331. doi: 10.1109/ICUAS.2016.7502664.

[138] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1947–1950, 2010. ISBN 9781424453085. doi: 10.1109/ISCAS.2010.5536970.

[139] Kirk Y.W. Scheper and Guido C.H.E. de Croon. Evolution of robust high speed optical-flow-based landing for autonomous mavs. *Robotics and Autonomous Systems*, 124:103380, 2020. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2019.103380.

[140] Stefan Schliebs and Nikola Kasabov. *Computational Modeling with Spiking Neural Networks*, pages 625–646. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-30574-0. doi: 10.1007/978-3-642-30574-0_37. URL https://doi.org/10.1007/978-3-642-30574-0_37.

[141] Juergen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61: 85–117, 04 2014. doi: 10.1016/j.neunet.2014.09.003.

[142] Benjamin Schrauwen and Jan Van Campenhout. Improving spikeprop: Enhancements to an error-backpropagation rule for spiking neural networks. In *Proceedings of the 15th ProRISC Workshop*, pages 1–1, 2004.

[143] M. Schuetz, M. Oesterlein, C. Birkenhauer, and M. Vossiek. A custom lightweight uav for radar remote sensing: Concept design, properties and possible applications. In *2017 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 107–110, 2017. ISBN 9781509043545. doi: 10.1109/ICMIM.2017.7918868.

[144] Gene I. Sher. Handbook of neuroevolution through erlang. *Handbook of Neuroevolution Through Erlang*, pages 1–831, 1 2013. doi: 10.1007/978-1-4614-4463-3.

[145] A. Shrestha, K. Ahmed, Y. Wang, and Q. Qiu. Stable spike-timing dependent plasticity rule for multilayer unsupervised and supervised learning. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1999–2006, 2017. ISBN 9781509061815. doi: 10.1109/IJCNN.2017.7966096.

[146] Michael J. Skocik and Lyle N. Long. On the capabilities and computational costs of neuron models. *IEEE Transactions on Neural Networks and Learning Systems*, 2014. ISSN 21622388. doi: 10.1109/TNNLS.2013.2294016.

[147] M.I. Skolnik. *Introduction to Radar Systems*. Electrical engineering series. McGraw-Hill, 2001. ISBN 9780071181891. URL https://books.google.nl/books?id=Y6-APwAACAAJ.

[148] Steven Skorheim, Peter Lonjers, and Maxim Bazhenov. A spiking network model of decision making employing rewarded stdp. *PLOS ONE*, 9(3):1–15, 03 2014. doi: 10.1371/journal.pone.0090821. URL https://doi.org/10.1371/journal.pone.0090821.

[149] M. Spüler, S. Nagel, and W. Rosenstiel. A spiking neuronal model learning a motor control task by reinforcement learning and structural synaptic plasticity. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015. ISBN 9781479919604. doi: 10.1109/IJCNN.2015.7280521.

[150] Jan Stuijt, Manolis Sifalakis, Amirreza Yousefzadeh, and Federico Corradi. μBrain: An Event-Driven and Fully Synthesizable Architecture for Spiking Neural Networks. *Frontiers in Neuroscience*, 15:538, 2021. ISSN 1662-453X. doi: 10.3389/fnins.2021.664208.

[151] Sohan Suvarna, Dibyayan Sengupta, Pavankumar Koratikere, and Rajkumar S. Pant. Simulation of autonomous airship on ros-gazebo framework. *2019 Fifth Indian Control Conference (ICC)*, pages 237–241, 2019.

[152] V. Sze, Y. Chen, T. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[153] Amirhossein Tavanaei and Anthony S. Maida. Multi-layer unsupervised learning in a spiking convolutional neural network. In *Proceedings of the International Joint Conference on Neural Networks*, 2017. ISBN 9781509061815. doi: 10.1109/IJCNN.2017.7966099.

[154] Simon Thorpe and Jacques Gautrais. *Rank Order Coding*, pages 113–118. Springer US, Boston, MA, 1998. ISBN 978-1-4615-4831-7. doi: 10.1007/978-1-4615-4831-7_19. URL https://doi.org/10.1007/978-1-4615-4831-7_19.

[155] Simon Thorpe, Denis Fize, and Catherine Marlot. Speed of processing in the human visual system. *Nature*, 381(6582):520–522, 1996. doi: 10.1038/381520a0. URL https://doi.org/10.1038/381520a0.

[156] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. Spike-based strategies for rapid processing. *Neural networks : the official journal of the International Neural Network Society*, 14(6-7):715—725, 2001. ISSN 0893-6080. doi: 10.1016/s0893-6080(01)00083-1. URL https://doi.org/10.1016/s0893-6080(01)00083-1.

[157] Simon J. Thorpe, Rudy Guyonneau, Nicolas Guilbaud, Jong-Mo Allegraud, and Rufin VanRullen. Spikenet: real-time visual processing with one spike per neuron. *Neurocomputing*, 58-60:857 – 864, 2004. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2004.01.138. URL http://www.sciencedirect.com/science/article/pii/S0925231204001432. Computational Neuroscience: Trends in Research 2004.

[158] Peter Tiňo and Ashley Mills. Learning beyond finite memory in recurrent networks of spiking neurons. In Lipo Wang, Ke Chen, and Yew Soon Ong, editors, *Advances in Natural Computation*, pages 666–675, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31858-3.

[159] Vera Tyrsa, Adrian Carrio, Carlos Sampedro, Alejandro Rodriguez-Ramos, and Pascual Campoy. A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors*, 2017:3296874, 2017. doi: 10.1155/2017/3296874. URL https://doi.org/10.1155/2017/3296874.

[160] Fawwaz Ulaby and Long. *Microwave Radar and Radiometric Remote Sensing*. 2014. doi: 10.3998/0472119356.

[161] Sergio Valadez-Godínez, Humberto Sossa, and Raúl Santiago-Montero. On the accuracy and computational cost of spiking neuron implementation. *Neural Networks*, 2020. ISSN 18792782. doi: 10.1016/j.neunet.2019.09.026.

[162] S. Van der Zwaan, A. Bernardino, and J. Santos-Victor. Vision based station keeping and docking for an aerial blimp. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, volume 1, pages 614–619, 2000. doi: 10.1109/iros.2000.894672.

[163] Rufin Van Rullen, Jacques Gautrais, Arnaud Delorme, and Simon Thorpe. Face processing using one spike per neurone. In *BioSystems*, 1998. doi: 10.1016/S0303-2647(98)00070-7.

[164] C. Waldschmidt and H. Meinel. Future trends and directions in radar concerning the application for autonomous driving. In *2014 11th European Radar Conference*, pages 416–419, 2014. ISBN 9782874870378. doi: 10.1109/EuRAD.2014.6991296.

[165] Changhuang Wan, Nathaniel Kingry, and Ran Dai. *Design and Autonomous Control of a Solar-Power Blimp*. ISBN 9781624105265. doi: 10.2514/6.2018-1588. URL https://arc.aiaa.org/doi/abs/10.2514/6.2018-1588.

[166] Lei Wang, Simon X. Yang, and Mohammad Biglarbegian. Bio-inspired navigation of mobile robots. In Mohamed Kamel, Fakhri Karray, and Hani Hagras, editors, *Autonomous and Intelligent Systems*, pages 59–68, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-31368-4. doi: 10.1007/978-3-642-31368-4_8.

[167] Xiangwen Wang, Xianghong Lin, and Xiaochao Dang. Supervised learning in spiking neural networks: A review of algorithms and evaluations. *Neural Networks*, 125:258–280, 2020. ISSN 0893-6080.

[168] Xiuqing Wang, Zeng-Guang Hou, Feng Lv, Min Tan, and Yongji Wang. Mobile robots′ modular navigation controller using spiking neural networks. *Neurocomputing*, 134:230 – 238, 2014. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2013.07.055. URL http://www.sciencedirect.com/science/article/pii/S0925231214000976. Special issue on the 2011 Sino-foreign-interchange Workshop on Intelligence Science and Intelligent Data Engineering (IScIDE 2011) Learning Algorithms and Applications.

[169] Y. Wang, G. Zheng, D. Efimov, and W. Perruquetti. Improved altitude control method with disturbance compensation for an indoor blimp robot. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 3902–3907, 2017. ISBN 9781509028733. doi: 10.1109/CDC.2017.8264233.

[170] Yue Wang, Gang Zheng, Denis Efimov, and Wilfrid Perruquetti. Altitude Control for an Indoor Blimp Robot. *IFAC-PapersOnLine*, 2017. ISSN 24058963. doi: 10.1016/j.ifacol.2017.08.1909.

[171] K. Watanabe, N. Okamura, and I. Nagai. Closed-loop control experiments for a blimp robot consisting of four-divided envelopes. In *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, pages 2568–2573, 2015. ISBN 9781479917624.

[172] Q Wu, TM McGinnity, LP Maguire, R Cai, and M Chen. A visual attention model based on hierarchical spiking neural networks. *Neurocomputing*, 116:3–12, 2013. ISSN 0925-2312. doi: 10.1016/j.neucom.2012.01.046.

[173] QingXiang Wu, Martin McGinnity, Liam Maguire, Ammar Belatreche, and Brendan Glackin. Edge detection based on spiking neural network model. In De-Shuang Huang, Laurent Heutte, and Marco Loog, editors, *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, pages 26–34, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74205-0.

[174] QingXiang Wu, T. M. McGinnity, Liam Maguire, G. D. Valderrama-Gonzalez, and Patrick Dempster. Colour image segmentation based on a spiking neural network model inspired by the visual system. In De-Shuang Huang, Zhongming Zhao, Vitoantonio Bevilacqua, and Juan Carlos Figueroa, editors, *Advanced Intelligent Computing Theories and Applications*, pages 49–57, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-14922-1.

[175] J. N. Yasin, S. A. S. Mohamed, M. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila. Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches. *IEEE Access*, 8:105139–105155, 2020.

[176] Jian Zhang, Jianbo Hu, Juyu Lian, Zongji Fan, Xuejun Ouyang, and Wanhui Ye. Seeing the forest from drones: Testing the potential of lightweight drones as a tool for long-term forest monitoring. *Biological Conservation*, 2016. ISSN 00063207. doi: 10.1016/j.biocon.2016.03.027.

[177] Feifei Zhao, Yi Zeng, and Bo Xu. A brain-inspired decision-making spiking neural network and its application in unmanned aerial vehicle. *Frontiers in Neurorobotics*, 12:56, 2018. ISSN 1662-5218.

[178] Jean-Christophe Zufferey, Alexis Guanella, Antoine Beyeler, and Dario Floreano. Flying over the reality gap: From simulated to real indoor airships. *Autonomous Robots*, 21(3): 243–254, 2006. doi: 10.1007/s10514-006-9718-8. URL https://doi.org/10.1007/s10514-006-9718-8.