NEUROMORPHIC
Computing and Engineering

PAPER

CrossMark

OPEN ACCESS

RECEIVED
31 March 2022

REVISED
4 July 2022

ACCEPTED FOR PUBLICATION
6 July 2022

PUBLISHED
5 August 2022

Neuromorphic computing for attitude estimation onboard quadrotors

Stein Stroobants*, **Julien Dupeyroux** and **Guido C H E de Croon**

Micro Air Vehicle Lab, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands
* Author to whom any correspondence should be addressed.

E-mail: s.stroobants@tudelft.nl

Keywords: neuromorphic computing, spiking neural networks (SNN), unmanned aerial vehicles (UAVs), micro air vehicles (MAVs), neuromorphic processor

Abstract

Compelling evidence has been given for the high energy efficiency and update rates of neuromorphic processors, with performance beyond what standard Von Neumann architectures can achieve. Such promising features could be advantageous in critical embedded systems, especially in robotics. To date, the constraints inherent in robots (e.g., size and weight, battery autonomy, available sensors, computing resources, processing time, etc), and particularly in aerial vehicles, severely hamper the performance of fully-autonomous on-board control, including sensor processing and state estimation. In this work, we propose a spiking neural network capable of estimating the pitch and roll angles of a quadrotor in highly dynamic movements from six-degree of freedom inertial measurement unit data. With only 150 neurons and a limited training dataset obtained using a quadrotor in a real world setup, the network shows competitive results as compared to state-of-the-art, non-neuromorphic attitude estimators. The proposed architecture was successfully tested on the Loihi neuromorphic processor on-board a quadrotor to estimate the attitude when flying. Our results show the robustness of neuromorphic attitude estimation and pave the way toward energy-efficient, fully autonomous control of quadrotors with dedicated neuromorphic computing systems.

1. Introduction

Over the last two decades, efforts have been made to combine the fields of Artificial Intelligence (AI) and robotics with outstanding results [1]. Algorithms making use of AI techniques, such as deep neural networks, have been proposed to achieve state estimation [2], object manipulation [3], localization [4] and control [5–7]. For instance in [8], a quadrotor learns to fly by applying reinforcement learning (RL) to a densely connected neural network. Aerial vehicles are critical embedded systems, the constraints of which (e.g., size and weight, battery autonomy, sensors, computing resources) hamper the development and the performance of fully autonomous and robust on-board control. In particular, micro air vehicles (MAVs) are a class of aerial vehicles that could strongly benefit from AI-powered solutions to handle their highly non-linear dynamics, and allow for online adaptations to unpredictable changes occurring in the real world (e.g., gusts of wind, sensor damage, communication failure, etc). Many of the complex tasks future MAVs will have to perform will be powered by AI. One can think of deep neural networks that have to estimate optical flow [9] and recognize objects [10], which can be used for vision-based autonomous navigation in urban areas to deliver packages.

However, MAVs are now locked out from using large-scale neural networks because of the great amount of energy required, as well as disproportionate need for computing resources that only graphics processing units can offer. Additionally, standard Von Neumann architectures suffer from a relatively high latency that restrict their use in extreme conditions such as drone racing and aggressive flight maneuvers.

Alternatively to traditional artificial neural networks (ANNs) where the information is processed in a synchronous manner, spiking neural networks (SNNs) could represent the choice solution to bridge the gap between AI and resource-restricted robotics. In contrast to ANNs, SNNs encode the information not by the intensity of the signal, but by a series of binary events, also called spikes, and the relative time between them.

Inspired by their biological counterpart, spiking neurons accumulate incoming synaptic currents over time and fire whenever their membrane potential exceeds a certain threshold. While a wide range of neuron models have been proposed, the most commonly used are the integrate-and-fire and leaky-integrate-and-fire (LIF) [11]. The simpler binary signals and sparse firing of SNNs hold the promise of orders of magnitude more energy-efficient processing than ANNs [12].

The move toward SNNs requires a complete shift in the coding and processing of information that is not optimized for Von Neumann architectures. In this regard, neuromorphic sensors and processors have been designed, arousing the enthusiasm of roboticists to embed these new technologies onboard robots. Examples of neuromorphic processors include HICANN [13], NeuroGrid [14], IBM's TrueNorth [15], APT's SpiNNaker [16] which is part of the human brain project [17], and Intel's Loihi [18]. In terms of neuromorphic sensing, most efforts have been put to develop the neuromorphic equivalent to standard CMOS cameras, namely, event-based cameras [19]. Neuromorphic tactile sensors have also been proposed [20].

Tackling computationally expensive vision or navigation tasks with neuromorphic sensing and processing will bring large energy and speed benefits. However, to optimally reap the benefits of neuromorphic algorithms and hardware, an end-to-end, fully neuromorphic solution needs to be designed. Then a single neuromorphic processor could suffice. In the literature, some of the steps toward such a fully neuromorphic pipeline have been demonstrated. In [21], the authors introduced a neuromorphic PD controller that outperforms state-of-the-art controllers in high-speed control thanks to the synergy between the high update rates of the neuromorphic chip and the event-camera. Also, autonomous thrust control of a flying quadrotor was achieved by an SNN evolved in simulation, using optic flow to enact a constant-divergence landing [22]. In [23], the authors show that SNNs are capable of solving planning tasks, such as avoiding obstacles with a robotic arm.

This fully neuromorphic pipeline will also have to include 'lower-level' tasks, but these are not well studied until now. For example, autonomous control onboard MAVs requires an accurate estimate of the states, such as attitude and global lateral position and velocity, by combining sensor measurements. In this article, we propose a neuromorphic solution for onboard attitude estimation of a MAV using data from an inertial measurement unit (IMU), combining data from a three-axes accelerometer and a three-axes gyroscope. We compare the network to a similarly sized and trained traditional recurrent neural network (RNN) and commonly used filters specific to this task. The proposed SNN is trained from limited data obtained with a real MAV and can be employed as part of an autonomous neuromorphic flight-controller pipeline for an MAV. Closest to our work is the study in [24], which trains a traditional ANN with recurrency for attitude estimation. Specifically, in [24] it is shown that a two-layer RNN can be trained to perform this task with outstanding results on pre-gathered datasets. This network, however, is not spiking and has not been applied in the control loop of MAVs in flight.

MAVs are usually equipped with an IMU and use the combination of the angular velocities and the linear accelerations to estimate the current attitude. Angular velocities show the rate of change, but induce integration errors over longer time-windows while the linear acceleration shows the gravity vector over longer stretches of time. The output pitch and roll estimates are necessary for the drone to be able to control the position in the x–y plane by performing attitude control.

Our contributions are threefold. First, we propose an SNN architecture to perform state estimation for dynamic systems such as quadrotors with limited data obtained with a physical quadrotor and ground-truth. Then, we demonstrate that the proposed neuromorphic state estimator exhibits competitive performance when compared to widely used, non-neuromorphic solutions (i.e., Madgwick filter, Mahony filter, and complementary filter) and to a traditional RNN. Lastly, we successfully test our solution onboard a quadrotor equipped with the Loihi neuromorphic chip, thus paving the way for a fully neuromorphic control-loop for quadrotors.

2. Methods

2.1. Spiking network definition

This section introduces the attitude estimation SNN, called Att-SNN. The different components that make up the network are shown in figure 1.

2.1.1. Spiking neuron model

In this work, we use the LIF neuron as the core of our SNN. Widely used in the literature, the LIF model is available in most SNN simulators a neuromorphic hardware, including the Loihi used for this study [18]. The discrete-time difference equations governing the LIF neuron are given as follows:

$$v_i(t+1) = \tau_i^{\mathrm{mem}} v_i(t) + i_i(t) \tag{1}$$

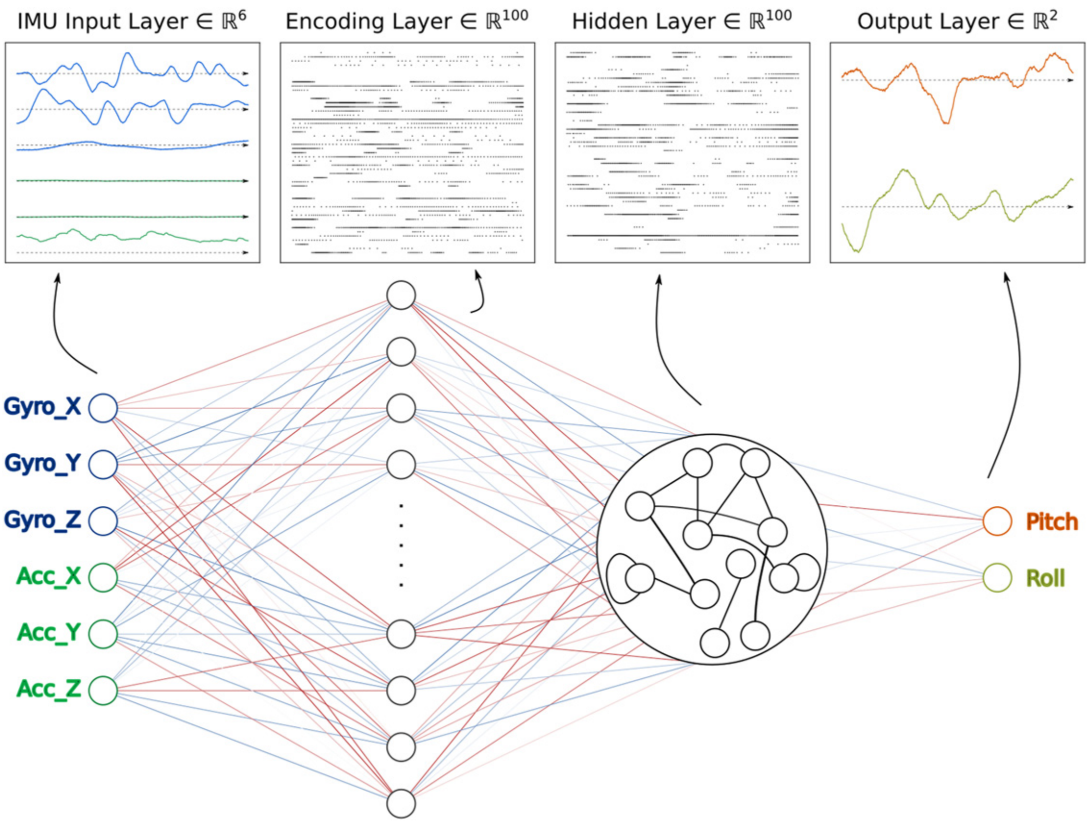

IMU Input Layer $\in \mathbb{R}^6$
Encoding Layer $\in \mathbb{R}^{100}$
Hidden Layer $\in \mathbb{R}^{100}$
Output Layer $\in \mathbb{R}^2$
Gyro_X
Gyro_Y
Gyro_Z
Acc_X
Acc_Y
Acc_Z
Pitch
Roll

Figure 1. Topology of the attitude estimation network Att-SNN showing signals transported between layers. In the encoding layer, the normalized IMU data is transformed into spikes. The next layer is a fully connected recurrent layer that sends spikes to the leaky-integrator (LI) output layer, converting spikes back to attitude estimates.

$$i_i(t+1) = \tau_i^{\text{syn}} i_i(t) + \sum w_{ij} s_j(t) \tag{2}$$

where $\upsilon_i(t)$ is the membrane potential at time t, $\tau_i^{\text{mem}} \in [0, 1]$ and $\tau_i^{\text{syn}} \in [0, 1]$ the membrane and synaptic time constants, $i(t)$ the synaptic current at time t, w_{ij} the synaptic weight between neurons i and j, and s_j a binary value representing either a spike or no spike coming from the pre-synaptic neuron j. To determine whether a neuron emits a spike, the membrane potential is reduced with the neurons firing threshold θ_i^{thr} and passed through the Heaviside step-function to determine the output of the neuron:

$$s_i(t) = H(\upsilon_i(t) - \theta_i^{\text{thr}}) = \begin{cases} 0, & \upsilon_i(t) - \theta_i^{\text{thr}} \leqslant 0 \\ 1, & \upsilon_i(t) - \theta_i^{\text{thr}} > 0 \end{cases}. \tag{3}$$

When the Heaviside-function resolves to 1 and the neuron emits a spike, the membrane potential $\upsilon_i(t)$ is reset to zero.

2.1.2. Data encoding

Standard, off-the-shelf IMUs are not neuromorphic; the output data is formatted as floating point values and streamed synchronously. As a result, the measured angular rates and linear accelerations must be translated into spikes so that they can be processed by the SNN.

Data encoding is a complex task. Spike coding algorithms can be divided into three categories. Population coding uses set of distinct neurons to encode (or decode) the signal by emitting a spike whenever the input signal falls within the range distribution of one (or several) neuron. It has been successfully applied in [22, 25]. Temporal coding algorithms encode the information with high timing precision, by emitting a spike whenever the variation of the input signal exceeds a threshold. Rate coding, which was used in [23], encodes the information into the firing frequency of a population of neurons.

In this work, the floating point values returned by the 6-DOF IMU are encoded into spikes by means of a spiking layer densely connected to the six outputs of the IMU sensor (figure 1). The synaptic weights, as well as the dynamics of the neurons in this encoding layer were learned offline. Both the time-constants were trained for each neuron separately, resulting in $2N$ parameters, with N the number of neurons in the encoding layer.

Before passing the IMU data to the network, a normalization process is applied to ensure faster learning and convergence during training. This normalization is achieved through min–max, mapping the input data to a range of $[-1, 1]$, according to the formula hereafter. The values of $x_{\min}$ and $x_{\max}$ were chosen empirically to match the full range of possible inputs.

$$x_{\text{norm}}(t) = 2 \cdot \frac{x(t) - x_{\min}}{x_{\max} - x_{\min}} - 1. \tag{4}$$

2.1.3. Decoding the global attitude

Decoding of the spiking activity to an estimate of the attitude in radians is performed via a non-spiking decoding layer (figure 1), composed of two LI neurons. In this case, the membrane potential $v_i(t)$ in equation (1) of the neuron is directly used as the output of the layer, providing a stateful decimal value representing the current pitch or roll attitude. In contrast to the input-data, no further normalization is required for the attitude values. The pitch and roll angles in the training data, expressed in radians, are already distributed around a mean of zero with standard deviation (SD) of 0.25. Theoretically, the limits go from $-\pi$ to π, but since we are targeting a quadrotor in a normal flight regime around hover, these values will not be reached. Lastly, since the pitch and roll axes of the MAV are symmetric (this is true in the case of a standard, symmetric quadrotor), the neuron parameters of the decoding layer are set equal during training. This results in training two parameters for this layer.

2.2. Training

2.2.1. Training setup

The proposed Att-SNN was trained both in simulation, using datasets created with the RotorS simulator [26] (figure 2(a)), and with data collected with a quadrotor flying in the real world[1] (figure 2(b)). The quadrotor is equipped with a Pixhawk 4 Mini flight controller, which combines the measurement of two separate IMU's for redundancy. The IMU data are logged at 200 Hz. For these experiments, the ground truth was provided by a multi-camera motion capture system (OptiTrack). This system provides sub-degree accuracy attitude estimates at the same rate as the combined IMU measurements from the quadrotor. An overview of the distribution of the IMU and OptiTrack data collected both in simulation and in the real world is provided in figure 3. In total, 35 datasets of 100 s were gathered in simulation and 11 datasets of 100 s in our real-world tests. This amounts to a total of 77 min of flight time. Both the simulation and real-world data sets have been gathered in a way as to represent roll and pitch angles between hover and swift flight, reaching relatively large angles up to 45 degrees. This covers a normal flight regime. An important challenge in IMU-based attitude estimation is that the gyros and accelerometers have biases that can change over time. For instance, accelerometers usually suffer from a turn-on bias which shows as a constant offset of the measured acceleration. This bias is especially visible in the x-axis of the PX4 accelerometer. It is also worth noting that there is also a difference in the range of values in the $gyro_z$, which shows that the quadrotor in simulation was rotating more around the z-axis than in the real-world experiments. Since this rotation is only affecting the yaw angles directly, the influence on the pitch and roll estimates is limited. The datasets were split up in 70% train and 20% validation, and 10% test. The training datasets were split into sequences of 10 s containing 2000 time steps. Furthermore, the simulated data was augmented by adjusting the accelerometer turn-on bias and both the accelerometer and gyroscope noise densities to reduce the reality gap. The noise characteristics were determined using the steady-state error of the real world data set.

The Att-SNN was implemented using the Norse [27] python library, based on PyTorch. The Adam optimizer [28] was used with a learning rate of 0.005, combined with the k-step ahead, one step back Lookahead [29] optimizer to speed up learning. For the Lookahead optimizer, the value of α was set at 0.5 and k at 6. All code to reproduce the results can be found in footnote 2.

Since the Heaviside function used in the neuron dynamics (equation (1)) is non-differentiable, a surrogate-gradient (SG) was chosen to enable backpropagation through time (BPTT). A summary of SGs can be found in [30]. In this study, we used the SuperSpike [31] with a width of 20 as it is a suitable option for supervised BPTT and is robust to changes in the input paradigm as is the case with our current-based input [32].

The error between the output and the target was characterized by the mean-squared-error (MSE) loss function where both the pitch- and roll-error were weighted evenly. The MSE for a sequence S is calculated as in equation (5), with $\hat{\theta}_k$ and θ_k^{gt} the estimated and ground-truth pitch angles at timestep k, and $\hat{\phi}_k$ and ϕ_k^{gt} the estimated and ground-truth roll angles.

[1] The dataset will be made available online upon publication.

[2] All code will be made available online upon publication.

(a) AscTec Hummingbird flying in open-source RotorS quadrotor simulator [26], used to gather supplementary simulation data for training.

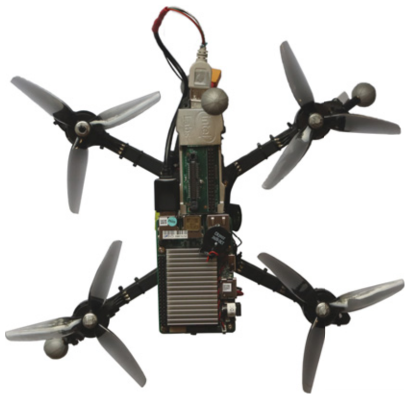

(b) Quadrotor used in our research to gather real-world data and for the evaluation of our SNN on the neuromorphic processor Loihi).

Figure 2. The quadrotors used for the simulation (RotorS Gazebo) and real-world tests (data acquisition and tests on the Loihi neuromorphic chip).

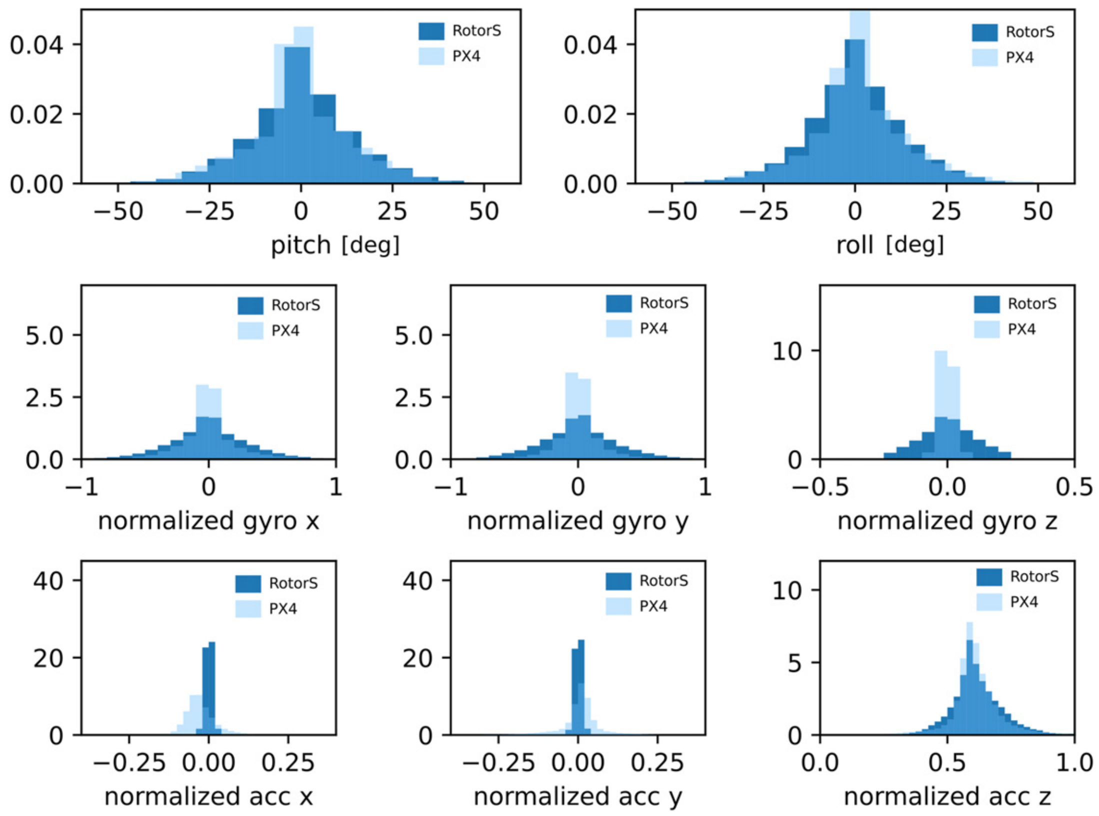


Figure 3. Distribution of the input–output data used for training the network. The inputs (lower six histograms) show sensor-readings after applying normalization.

$$\mathrm{MSE}_S = \sum_{k=0}^{N=\mathrm{len}(S)} \frac{(\hat{\theta}_k - \theta_k^{\mathrm{gt}})^2 + (\hat{\phi}_k - \phi_k^{\mathrm{gt}})^2}{2}. \tag{5}$$

The Att-SNN is trained with 100 neurons in the encoding layer and 100 in the hidden layer. The thresholds were fixed at 0.5 for all neurons. This results in 600 weights and 200 neuron parameters for the encoding layer, 10 000 + 10 000 weights and 200 neuron parameters for the hidden layer and 200 weights and two neuron parameters for the output layer. In total, this adds up to training of 20 800 weights and 402 neuron parameters. Every epoch, 15 batches of 40 sequences are randomly selected from all training data (including simulated and real-world data) for a training iteration, and afterwards the error is calculated for all validation sets. The training is stopped by a criterion based on a moving average of the error on the validation dataset over the last

20 epochs. If the moving-average of the error is higher than 110% of the lowest average validation loss so far, training is aborted. Besides, if the lowest validation loss so far did not change for at least 50 epochs, training is also aborted.

2.3. Implementation on neuromorphic hardware

To fully demonstrate the potential of SNNs for state estimation in MAV applications, the proposed Att-SNN architecture has been implemented on Intel's Loihi processor [18]. The constraints of SNN design imposed by the Loihi requires to adapt the SNN parameters, such as the synaptic weights and the neurons' parameters. A naive solution to this would be to quantize the parameters after training, but this would inevitably result in a loss in accuracy, and in the long run could affect the overall performance of the state estimation (and control). Alternatively, we have included the quantization in the training process by replacing the full-resolution weights with their quantized equivalents before the forward pass while propagating the gradients. By doing so, we ensure that the network converges to a solution that is fully compatible with the specific features of the Loihi chip. The quantization function is defined by:

$$p_q = \mathrm{round}(p/\Delta q)\Delta q, \quad p_q \in [q_{\min}, q_{\max}] \tag{6}$$

with p_q the quantized version of parameter p, Δq the quantization step-size, round(.) rounding of a floating point value to the closest integer and $[q_{\min}, q_{\max}]$ the quantization range. The threshold θ_{thr} was fixed during training, while all other parameters were not. The quantization ranges and step sizes used in this study are:

- Synaptic weights w_{ij}, range: $[-1, 1-\frac{1}{256}]$, step size: $\frac{2}{256}$
- Synaptic decay $\tau_{\mathrm{syn},i}$, range: $[0, 1]$, step size: $\frac{1}{4096}$
- Membrane decay $\tau_{\mathrm{mem},i}$, range $[0, 1]$, step size: $\frac{1}{4096}$.

During implementation on the neuromorphic chip, the parameters can be multiplied by the quantization range, resulting in integers compatible with the constraints set by the neuromorphic processor.

2.4. Models for comparison

The performance of the Att-SNN neuromorphic state estimator is compared to the most widely-used non-neuromorphic algorithms: (i) the Madgwick filter [33], (ii) the Mahony filter [34], the extended Kalman filter (EKF) [35, 36], and (iv) the complementary filter [37]. These filters all adequately estimate the pitch and roll angles of a quadrotor in flight. Since only data from a 6-DOF IMU is used, these two are the only observable states out of a full non-linear model of a quadrotor. For fairness of comparison, we implement a minimalistic EKF that uses the IMU together with a quaternion-based random walk model to estimate pitch and roll. Typically, EKFs are more extensive, as they can estimate more states such as velocity or position, integrate other sensors like GPS, and can employ more detailed motion models using known control inputs. However, such extensions are out of the scope of the current study.

These filters all require some fine-tuning of their parameters with respect to the dynamic motions captured in the datasets, the noise level measured in the IMU, and a prior knowledge of the initial state. Tuning these parameters by hand is quite common, but this may have resulted in unfair comparison with a neural network that is automatically and thoroughly trained with dedicated algorithms on the training dataset consisting of simulation and PX4 data. As a result, the parameters of all filters were automatically determined for the same datasets using the modified particle swarm optimization (PSO) algorithm as described in [38]. The optimization values were chosen as: $w = 0.8$, $c_1 = 0.15$ and $c_2 = 0.05$ and a total of 100 particles was used. The cost function was defined as the MSE per timestamp, averaged over all training sequences plus a high cost of 10 for parameters below 0 or above 1 to constrain the parameter to the corresponding ranges. Using the PSO, the optimal parameters were estimated for both the simulation and PX4 datasets, assuming that the filters had no knowledge of the initial angle. The impact of this assumption will be further analysed in section 3.1.

Additionally, we propose a traditional RNN, called Att-RNN which is composed of gated recurrent units (GRUs). This network has a very similar structure to our proposed Att-SNN, allowing to evaluate the difference in performance caused by the introduction of the LIF model's neural dynamics and spiking.

2.4.1. The complementary filter

The complementary filter is a widely used filter for attitude estimation due to its simplicity. The output of the filter is a weighted average between the angle θ_k^{acc} measured by the gravity vector measured with the accelerometer (when the assumption of weak-acceleration relative to gravity holds), and the previous estimated angle $\hat{\theta}_{k-1}$ propagated with the angular velocity ω_k^{gyr} measured by the gyroscopes. For a single axis this is defined as follows:

$$\hat{\theta}_k = \gamma(\hat{\theta}_{k-1} + \omega_k^{\mathrm{gyr}}\Delta t) + (1-\gamma)\theta_k^{\mathrm{acc}} \tag{7}$$

where γ is the weighting factor, balancing between the accelerometer that provides a solid baseline on a long time horizon and the gyroscope that is accurate for updating the angle on short time-scale, but suffers from integration errors on a longer scale.

2.4.2. The Mahony filter

Gyroscope measurements from low-cost IMUs can be biased, and the Mahony filter is an extension of the complementary filter that counters this [34]. By adding an integral term $\hat{b}_k$ to the filter, based on the error e_k between the angle measured by the accelerometer and the predicted angle, the gyroscope biases can be effectively canceled without increasing the computationally cost too much. The filter, in discrete quaternion form, can be written as:

$$\hat{q}_k = \hat{q}_{k-1} + \left(\frac{1}{2}\hat{q}_{k-1} \otimes \mathbf{p}\{w_k^{\text{gyr}} - \hat{b}_k + k_{\text{P}}e_k\}\right)\Delta t \tag{8}$$

$$\hat{b}_k = \hat{b}_{k-1} - k_{\text{I}}e_k\Delta t \tag{9}$$

Where $\mathbf{p}\{x\} = [0\,x]^{\text{T}}$ is a pure quaternion that relates to the rotation velocity of the attitude and $p \otimes q$ is the Hamilton product between quaternion p and q. This algorithm can be optimized for certain motions or sensors by adjusting the proportional (k_{P}) and integral (k_{I}) gain.

2.4.3. The Madgwick filter

Madgwick [33] defines attitude estimation as a minimization problem, solved with a gradient descent algorithm. While the gyroscope is still used for integrating the angle, the error with respect to measured gravity is represented as a cost function $f(q)$ describing the difference between the angle measured with the accelerometer and the gravity vector rotated to the body-frame. This cost-function is minimized by taking a single gradient descent step. This results in the following update step in discrete quaternion form:

$$\hat{q}_k = \hat{q}_{k-1} + \left(\frac{1}{2}\hat{q}_{k-1} \otimes \mathbf{p}\{w_k^{\text{gyr}}\} - \beta\frac{\frac{1}{2}\text{Jac}(f(\hat{q}_{k-1}))^{\text{T}}f(\hat{q}_{k-1})}{\|\frac{1}{2}\text{Jac}(f(\hat{q}_{k-1}))^{\text{T}}f(\hat{q}_{k-1})\|}\right)\Delta t \tag{10}$$

where $\text{Jac}(f(\hat{q}_{k-1}))$ is the Jacobian matrix of the cost-function evaluated at $\hat{q}_{k-1}$ and β the optimization parameter. Although this filter can be used for 6-DOF IMU data, it is typically used on 9-DOF IMUs that also have a tri-axial magnetometer. The output quaternions of both the Madgwick and Mahony filters will be transformed to Euler angles for comparison with the other methods.

2.4.4. The extended Kalman filter

The EKF is one of the most-used fusion algorithms for non-linear systems and can also be applied to attitude estimation with a 6-DOF IMU [39]. It uses a model of the system to predict the state at the next time-step and corrects this prediction with measurements using a prediction of the estimate covariance. This implementation uses the angular rate measured by the gyroscope in the prediction step and the angle measured from the accelerometer measurements in the correction step. Since noise on the dynamic model and sensor measurements is defined by covariance matrices, the balance between the gyroscope and accelerometer can be optimized by changing these matrices.

2.4.5. A non-neuromorphic neural network

It is not common to use ANNs for attitude estimation, although there is an increasing interest in using ANNs for low-level estimation and control tasks. Here, we also compare our neuromorphic algorithm with a more traditional neural network. As a comparison to the Att-SNN, it was decided to implement a RNN consisting of two layers with GRUs similar to the one proposed in [24]. However, in our implementation the network has Euler-angles in radians as outputs, instead of quaternions to keep it comparable to our Att-SNN, during training and evaluation. The recurrency and memory in the GRUs allow the network to store and keep track of the states in the network. This is something a regular feedforward ANN would not be able to do, but that is important for integrating and filtering information over time for state estimation. The network has the same number of neurons per layer as the Att-SNN and is trained using the same optimizer and training strategy. The differences lie in (i) the Att-ANN uses GRUs, while the Att-SNN does not, (ii) the neural dynamics, where the Att-ANN sets its state directly based on all feedforward and recurrent connections, and Att-SNN implements the LIF neuron model.

3. Results

In the following, we investigate the performance of the proposed Att-SNN. We first compare the accuracy of the Att-SNN with state-of-the-art, non-neuromorphic attitude estimation filters commonly in control of

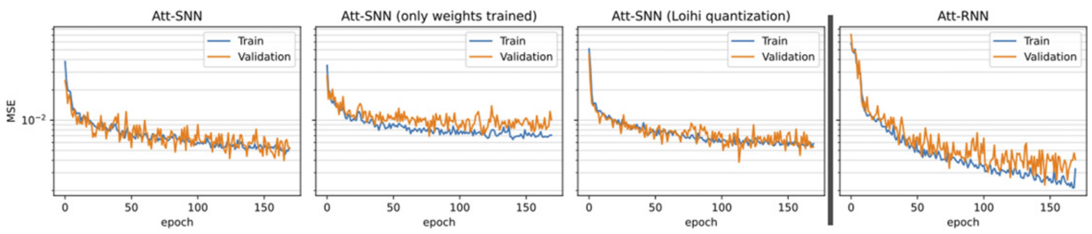


Figure 4. Training loss curve of the Att-SNN vs the similarly trained Att-RNN with GRUs on training and validation data.

Table 1. Results of the *k*-fold cross-validation showing the mean and SD on the test- and training datasets.

Fold	Train error		Test error	
	Mean	SD	Mean	SD
1	2.09	0.32	3.43	0.44
2	2.21	0.50	1.78	0.32
3	1.82	0.25	2.42	0.39
4	2.05	0.38	2.65	0.48
5	2.18	0.51	2.99	0.47
sim	1.97	0.23	4.86	0.80
PX4	2.20	0.41	5.40	0.65

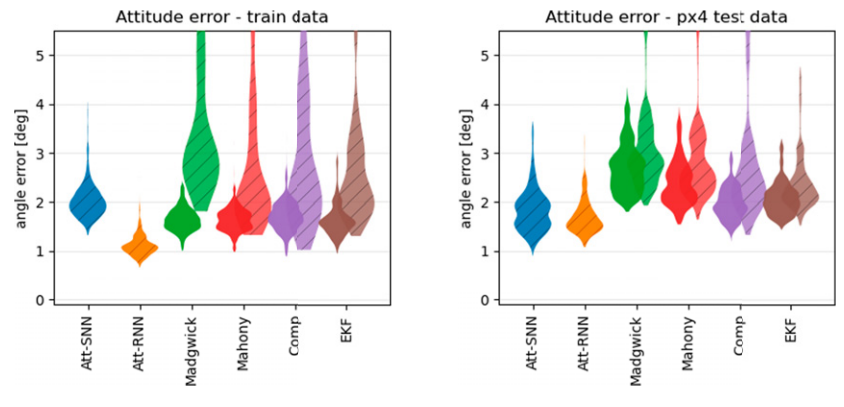
quadrotors. The spike activity of the Att-SNN is then evaluated with respect to the maneuvers performed by the quadrotor to better characterize the dynamic response of the Att-SNN. Lastly, the neuromorphic state estimator is implemented within the control loop to demonstrate the stability and robustness over time while unknown disturbances are applied to the MAV.

3.1. Performance analysis

First, we look at the performance of the neural network methods during training, where we compare three variants of the Att-SNN with the Att-RNN (figure 4). The Att-SNN and the Att-RNN models differ in two aspects. The training of the Att-RNN model results in a more important decrease of the loss function (MSE, equation (5)) than for the neuromorphic model. Moreover, the validation loss of the Att-SNN model reveals a higher stochasticity (SD of loss minus the moving average of 1.7×10^{-3} vs 0.9×10^{-3}) than that of the Att-RNN model, which can be explained by the non-differentiability of the Heaviside function used in the LIF model (equation (3)). Both networks suffer to some extent from overfitting, since the loss on the training data is lower than that on validation data. By using data with a wide range of dynamic motions and early-stopping during training, generalization can be improved. The influence of the dataset on the ability of the networks to generalize over data samples never seen during training can be identified by performing a *k*-fold test. We have split up all PX4 and simulation datasets in 70% train, 20% validation and 10% test data in five different folds. We have also included two folds where the network trained on simulation is tested on the PX4 data and vice versa. These are depicted as *sim* and *PX4*. The mean error and SD (table 1) for these folds on the corresponding test sets show that the model is resilient to the motions observed in the training data, and new motions in the test set. The error on the test dataset for the second fold is even lower than on the training sets. This can be caused by less dynamic maneuvers in the datasets that were kept apart for this specific fold. It also shows that training on a single-source dataset results in extreme generalization errors, indicating the importance of mixed-data.

To allow the comparison of performance between all filters, the average angle error between the filter and the ground truth was determined for 40 sequences spanning 50 s of real-world flight.

In figure 5, we show the overall results of the two neural-based models (Att-SNN and Att-RNN) along with the four filters (Madgwick, Mahony, complementary, and EKF) with respect to the training PX4 datasets and the validation PX4 datasets. Overall quantitative results are further detailed in table 2. The performance of the standard filters has been measured both with and without prior knowledge of the initial state. The error distribution of the Att-SNN model shows that the network is able to estimate the attitude of the quadrotor with an error comparable to the common filters. The neural-based methods perform better than the standard filters on the training set, when those filters do not have access to the initial state of the quadrotor (i.e., footnote ‘a’ in the table). On the test set, all four filters perform slightly worse as compared to the results obtained with the training set, showing that the test dataset poses certain challenges that are more difficult for the traditional



(a) Attitude error on real-world PX4 training dataset.

(b) Attitude error on real-world PX4 test dataset.

Figure 5. Error distribution of the Att-SNN versus other filters for data obtained in flight. The diagonally hatched plots show the error when the initial angle is not known. Both the Att-SNN and Att-RNN always start without knowledge of the initial angle. Results are given in degrees and combine the results for both the pitch and roll angles.

Table 2. Overall results obtained for all attitude estimation models, compared by means of median, mean and SD on training and test data. Standard filters include results for training and test with and without prior knowledge on the initial state. Results are expressed in degrees.

	Training set			Test set		
	Median	Mean	SD	Median	Mean	SD
Att-SNN[a]	1.90	1.56	0.24	2.79	2.79	0.37
Att-RNN[a]	**1.53**	1.91	0.28	2.45	2.45	0.35
Madgwick	2.79	2.81	0.36	2.89	2.92	0.41
Madgwick[a]	3.03	3.22	0.83	3.06	3.34	1.11
Mahony	2.56	2.57	0.36	2.53	2.54	0.33
Mahony[a]	2.69	2.85	0.74	2.59	2.84	0.89
Complementary	2.42	2.46	0.42	**2.13**	2.13	0.35
Complementary[a]	2.60	2.98	1.24	2.36	2.71	1.36
EKF	2.67	2.75	0.52	2.36	2.40	0.44
EKF[a]	2.65	2.71	0.51	2.49	2.64	1.00

[a]No knowledge of initial state.

filters to handle. As expected, the performance of the neural based solutions deteriorates slightly as they now face unknown data. This overfitting may be caused by unknown motions or noise characteristics in the test set and could be reduced by increasing the training data. Adding data from different quadrotors that have different dynamic properties and IMUs with different noise characteristics will increase the generalization properties of the network. However, the average error still remains stable, below 2.5° which is in the same range as the errors of the common filters. In figure 6, we further demonstrate the impact of the initial attitude on the response of the filters. Due to the symmetry of the quadrotor along the x and y axes, only the pitch results are presented. Whereas both the Att-SNN and Att-RNN are able to converge to the real angle in less than a second (200 samples), the three conventional filters have not fully converged after 7 s. Since the filters all balance between the accuracy on short time-scales given by the gyroscopes and the long time-scale reference of the accelerometer, a large offset at $t = 0$ can result in a long convergence. This balance is forged by the optimization parameters that were found with the PSO, which remain equal over the course of a sequence. These results suggest that the neural networks have learned a way to perform this balance in an adaptive way, trusting the angle θ_k^{acc} more when they largely contradict the current $\hat{\theta}_k$, but rely less on these estimates when the assumption of weak-acceleration does not hold. An implementation of such an adaptive mechanism for attitude estimation with a 9-DOF IMU with an adaptive EKF was shown in [40]. To support this hypothesis, the plain complementary filter is adjusted by an adaptive law, trusting more on θ_k^{acc} if the gyroscopes show low angular velocity. If $\|\omega_k^{\text{gyr}}\| < 0.1$, the complementary gain γ is increased by the difference between the estimated angle $\hat{\theta}_k$ and the angle θ_k^{acc} times a gain k_a chosen as 0.01. The result, as seen in figure 6, is that the complementary filter recovers faster from an initial offset. Although this does improve the convergence, it is still slower than the neural-based solutions, and is less responsive during aggressive maneuvers. However, it might give some insight in the way the neural-based solutions handle this.

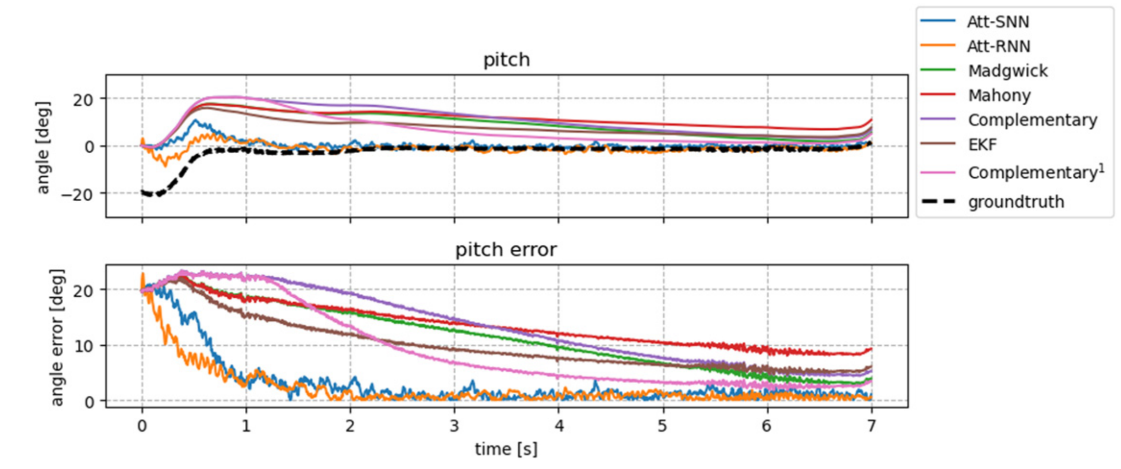
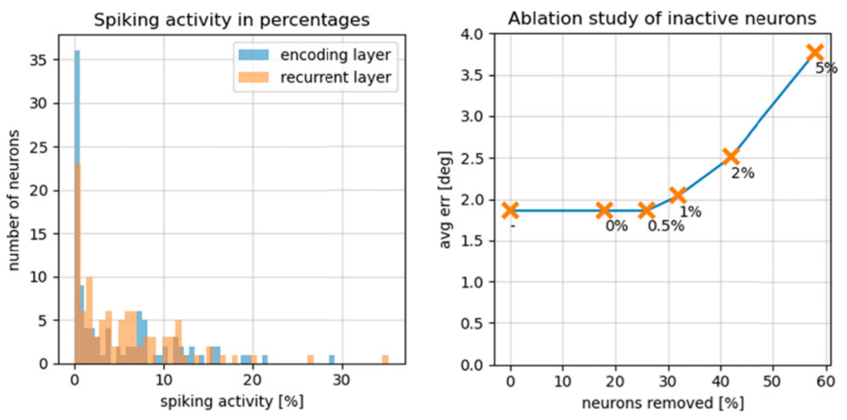


Figure 6. Pitch response and error for different filters w.r.t. ground truth for initial offset of $\approx 20^\circ$. In this case, all conventional filters are tested without any information about the initial angle. Real world PX4 data was used for this comparison. The complementary[a] shows the result of the complementary filter with an adaptation law to allow for quicker convergence.



(a) Histogram of the spiking activity in percentages of all neurons, averaged over the datasets

(b) Effect of removing sparsely spiking neurons on accuracy to reduce the overall network-size and prospective energy consumption in an end-to-end solution.

Figure 7. Investigation in the spiking activity of all neurons in the Att-SNN and the effect of removing sparsely spiking neurons.

3.2. Spiking activity

After training the Att-SNN model, we measured the overall spiking activity over the datasets of both the encoding and the hidden layers to evaluate the sparsity of the network. A histogram of the average spiking activity per neuron is given in figure 7(a). Results show that, out of all neurons, 27% of neurons in the encoding layer and 10% of neurons in the recurrent layer do not spike at all, and that an extra 7% and 8% respectively spike less than 0.5% of the time. To understand if these sparsely spiking neurons have a significant effect, an ablation study has been performed. By pruning all neurons that spike less than **x**% of the time for a subset of data, the effect on the accuracy for the rest of the data can be established. The results are shown in figure 7(b). Pruning the neurons that do not exceed an average spiking activity of 0.5% did not modify the performance of the network. This corresponds to a total of approximately 25% of the neurons of the Att-SNN model, which amounts to a reduction of almost half of the synapses while making the network more efficient.

3.3. Input dataset manipulation

In order to obtain some preliminary understanding of the functioning of the SNN, we have evaluated the fusion of sensory information by manipulating sensory inputs. In particular, we compared the output of the Att-SNN model during a constant velocity flight where the attitude is kept constant. In figure 8, the response of the network is provided for the pitch angle of the quadrotor obtained in simulation and in the following conditions: (i) the gyroscope is removed, (ii) the accelerometer is removed, (iii) the accelerometer data all point toward gravity in the world frame (resulting in $\theta_k^{\mathrm{acc}} = 0$), and (iv) no manipulation (i.e., the initial Att-SNN).

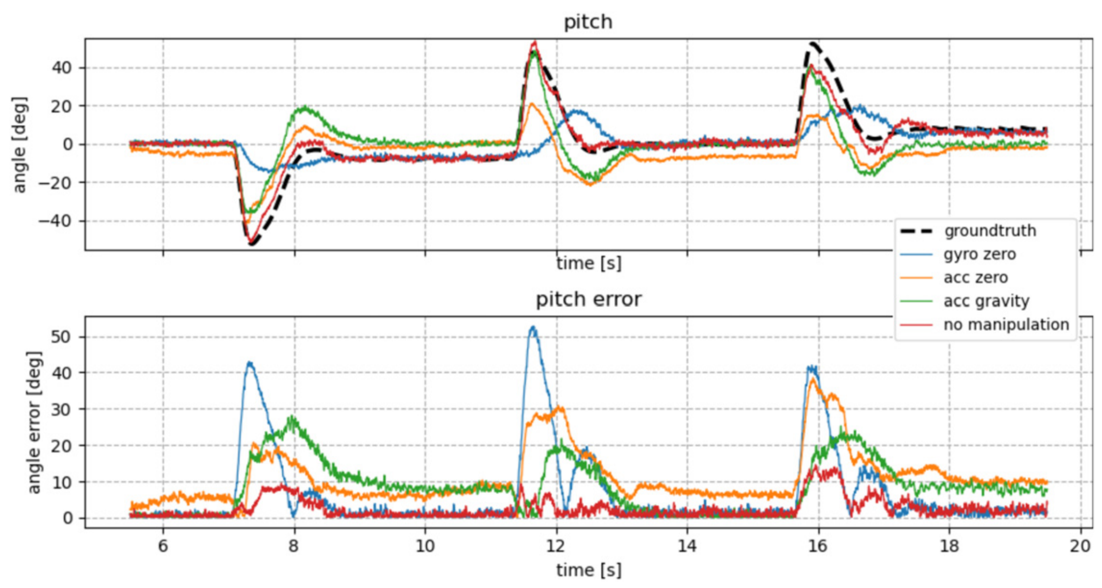
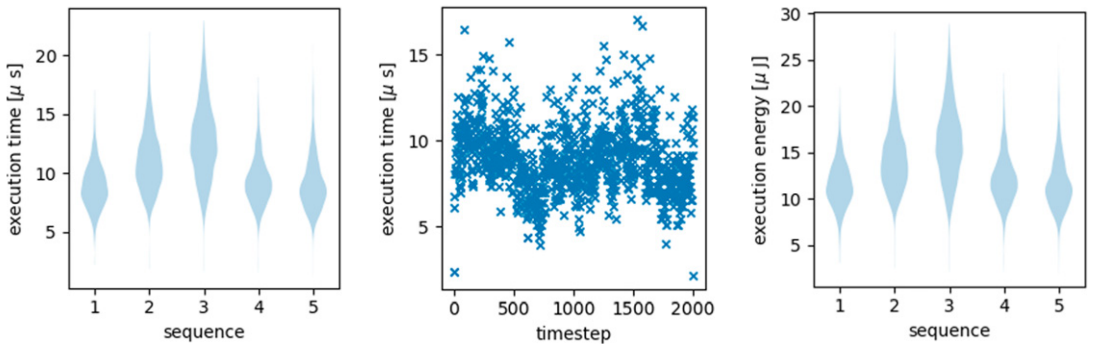


Figure 8. Response of the SNN for different types of input manipulation. Both the pitch estimation and error of the network is shown for (i) no data manipulation, (ii) all gyroscope values zero, (iii) all accelerometer values zero, and (iv) all accelerometer values pointing toward gravity in world-frame ($\theta_k^{\mathrm{acc}} = 0$).



(a) Execution time for 5 sequences of 2000 time steps (b) Execution times for an example sequence (c) Energy consumption for 5 sequences of 2000 time steps

Figure 9. Energy and evaluation frequency of the Att-SNN on the Nahuku board that has 32 Loihi chips.

We observe that removing the angular velocity information provided by the gyroscope results in large errors during fast motion, but remains correct when the motion follows a constant angle. This shows that the Att-SNN model uses the accelerometer data for a long time scale estimation of the angle, similar to how the common filters use it. When the accelerometer data is replaced by the gravity vector in the world frame, effectively measuring a zero angle at all times, this results in an estimate that is quite reliable during fast maneuvers. However, during constant velocity flight the output of the estimator is approximately zero, since it has no absolute measurement of the real angle. This shows that the network uses the accelerometer as an absolute reference to the global attitude, just like the comparison filters do.

3.4. Energy consumption and evaluation frequency on neuromorphic hardware

To evaluate the potential of deploying these networks on neuromorphic processors, the update frequency and energy consumption of the network were examined. Since gathering energy benchmarks on the Kapoho Bay, used on our quadrotor, is not possible, the results shown in figure 9 are obtained with a Nahuku board that has 32 Loihi chips by utilizing the energy and execution time probes as provided in the NxSDK. The average execution time of the spiking recurrent layer on hardware was only 10 μs, which corresponds to 100 000 Hz. As comparison, the execution time of the layer was also evaluated using a PyTorch implementation on a laptop with a Nvidia GTX 1650Ti and a 12-core Intel i7-10750H CPU. The average execution time for a single timestep of the Att-SNN here was 555 μs, $\approx$55 times as high as running the network on the neuromorphic processor. Also the execution time of the Att-RNN was measured in the same way, and the average execution time here was

150 μs, still an order of magnitude higher than on the neuromorphic processor. Also the number of floating-point operations (FLOPS) per method per timestep have been calculated. The Att-SNN requires only ($M^2 + MN + 2M$) FLOPS, with M the number of neurons in the hidden layer, and N the amount of inputs to this layer. The Att-RNN with gated-recurrent units requires approximately ($3M^2 + 3MN + 3M$) FLOPS for a single hidden layer, which is almost three times more. The average energy consumed per time step was 13.2 μJ. In our hardware architecture with the PX4 mini flight-computer connected to an UP Squared computer using Fast-RTPS, we were able to run it at the frequency of incoming IMU data, 200 Hz. It should be noted this looprate is dependent on the data transferring in the Von Neumann processors. In an end-to-end neuromorphic pipeline, this could be orders of magnitude faster. Since we run the Loihi at this frequency, the latency from input sample to estimate is governed by the delays in the network. In our implementation, this means an estimate is produced after three timesteps, resulting in a latency of 15 ms.

4. Discussion

This work demonstrates that a small recurrent SNN is able to estimate the attitude of a quadrotor in flight. In terms of accuracy, it is competitive with filters commonly used in flight-controllers, such as the Mahony-filter. Our approach is not data hungry, as common AI algorithms are, and only requires data obtained in simulation (easily accessible, and available open-source) supplemented with short sequences obtained with a real quadrotor (for our training, less than 20 min of real data was used). Our experiments have shown that little data already suffices for successful attitude estimation in nominal flight conditions. However, future work could consider larger datasets with a larger variety of flight maneuvers, and naturally a larger range of sensor biases. This would help to find out how well trained networks generalize to unseen conditions.

These results are obtained with focus on achieving a small-scale network. By increasing the size of the network, the training performance can be increased but the network might suffer more from overfitting. Already in our limited sized network, it was visible that training allowed for overfitting on the training data. This effect might be limited by supplying the training with more samples from different real world quadrotors or making the network more resilient to biases in the input (such as constant offsets of the measured accelerations) by adding parametrization. This parametrization could be obtained by adding adaptivity to the synaptic connections, for which a suitable online learning rule could be designed. Besides, minimizing size is in line with our goals of obtaining miniature-scale robots, capable of performing autonomous missions. Increasing the size too much will also constrain the learning, due to an increased dimensionality, resulting in high memory usage and training time.

A main potential criticism on our work could be that the plain complementary filter, achievable with only a couple lines of code in a microcontroller, is still a valuable choice for performing the task of attitude estimation based on IMU-data. Executed on a widely available and cheap micro-controller, it estimates states fast and with little energy expenditure. However, we aim for a fully neuromorphic pipeline, so that all processing, from very intensive visual processing to less intensive state estimation and control, can happen on a single neuromorphic chip. Hence, we also need to perform attitude estimation with an SNN. Moreover, from a scientific viewpoint, we are interested in understanding how (spiking) neural networks solve this task, potentially unveiling new strategies or delivering new hypotheses on sensor fusion in flying animals like small insects. Already in this study, the SNN showed the interesting property of converging quicker to the attitude when not initialized at the ground-truth attitude. In future work, we plan to delve deeper into the detailed workings of SNNs estimating attitude.

Currently, the work has the limitation that the Att-SNN only estimates the angles necessary for position control, and excludes yaw. Synthesizing an unbiased, non-diverging yaw-estimate using angular velocities and linear accelerations exclusively is not possible, so extra sensors would need to be included. A common option is the 3-DOF magnetometer, measuring the Earth's magnetic field and therefore supplying the estimator with an absolute measurement of the quadrotors heading. Although these sensors are very useful in large drones flying at great altitudes in non-urban environments, the readings are heavily influenced by disturbances in the magnetic field. These disturbances can be caused by electronic appliances in the vicinity or even by the motors of the quadrotor itself. In future work, adding data from other sensors or the output of a controller will be studied to capture a full 3D model of a quadrotor in flight. This will pose interesting new challenges such as dynamics changing over a flight because of a draining battery that might be addressed by an SNN that features adaptivity.

The frequency of the network in this study was chosen to be 200 Hz, since this matched the output of the sensor data from the PX4 flight-controller. In operation, the input-data streamed to the network has to be of the same rate as during training, because of the time-constants that characterize the neuron dynamics. During this research, it was found that training on even higher data rates, further reduced the estimation-error.

Together with the promises of extremely high update rates of neuromorphic hardware, this is encouraging for further research.

Considering the recurrent layer encodes the attitude (or rate of change of the attitude) necessary for control, the decoding layer can be replaced by another spiking layer that can be trained to perform control tasks, either by supervised learning mimicking a baseline controller, or it can be trained in a RL framework, as is discussed in [8]. These are the next steps that follow naturally from our research.

5. Conclusion

In this paper, we have presented an SNN model, called Att-SNN, that can be employed as an attitude estimator in an end-to-end neuromorphic control pipeline. This implementation builds upon three contributions. First, we have shown that the Att-SNN can perform state estimation tasks in highly dynamic systems such as MAVs, with competitive performance when compared to state-of-the-art non-neuromorphic methods and conventional recurrent ANNs. Second, we successfully implemented the Att-SNN on the Loihi neuromorphic processor, showing outstanding energy and time efficiency, therefore paving the way towards fully embedded neuromorphic control onboard MAVs. Third, this study shows for the first time that these networks can be used as an estimator in the control-loop, showing that small errors do not accumulate over time. Furthermore, our work shows an efficient method of encoding floating point sensor data into binary spikes without having to study the tuning curves of rate-coded neurons or finding the optimal distribution of neurons in population coding. Several prospective future research directions are recognized. This includes extending the estimation with neuromorphic control and reducing the effect of uncertainties due to sensor noise by employing online adaptivity. All together, the Att-SNN neuromorphic attitude estimation model will help closing the neuro-biologically-inspired control loop from sensor to actuator in critical embedded systems such as MAVs.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under Award Number FA8655-20-1-7044. The authors also wish to express their thanks to J Hagenaars and F Paredes-Valles for their fruitful discussions regarding implementing a Loihi-ready quantized version of the network.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: available at publication.

ORCID iDs

Stein Stroobants https://orcid.org/0000-0001-5733-1677
Julien Dupeyroux https://orcid.org/0000-0002-7414-5021
Guido C H E de Croon https://orcid.org/0000-0001-8265-1496

References

[1] Rajan K and Saffiotti A 2017 Towards a science of integrated AI and robotics *Artif. Intell.* **247** 1–9
[2] Ha D and Schmidhuber J 2018 Recurrent world models facilitate policy evolution *Advances in Neural Information Processing Systems* vol 31
[3] Agrawal P, Nair A V, Abbeel P, Malik J and Levine S 2016 Learning to poke by poking: experiential learning of intuitive physics *Advances in Neural Information Processing Systems* vol 29
[4] Giusti A *et al* 2016 A machine learning approach to visual perception of forest trails for mobile robots *IEEE Robot. Autom. Lett.* **1** 661–7
[5] Loquercio A, Kaufmann E, Ranftl R, Müller M, Koltun V and Scaramuzza D 2021 Learning high-speed flight in the wild *Sci. Robot.* **6** eabg5810
[6] Poznyak A S, Sanchez E N and Yu W 2001 *Differential Neural Networks for Robust Nonlinear Control: Identification, State Estimation and Trajectory Tracking* (Singapore: World Scientific)
[7] Lillicrap T P, Hunt J J, Pritzel A, Heess N M O, Erez T, Tassa Y, Silver D and Wierstra D 2016 Continuous control with deep reinforcement learning (arXiv:1509.02971)
[8] Hwangbo J, Sa I, Siegwart R and Hutter M 2017 Control of a quadrotor with reinforcement learning *IEEE Robot. Autom. Lett.* **2** 2096–103
[9] Ilg E, Mayer N, Saikia T, Keuper M, Dosovitskiy A and Brox T 2017 Flownet 2.0: evolution of optical flow estimation with deep networks *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition* pp 2462–70

[10] Redmon J, Divvala S, Girshick R and Farhadi A 2016 You only look once: unified, real-time object detection *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition* pp 779–88
[11] Kasabov N K 2019 Time-space, spiking neural networks and brain-inspired Artificial Intelligence available: http://springer.com/series/15821
[12] Cao Y, Chen Y and Khosla D 2015 Spiking deep convolutional neural networks for energy-efficient object recognition *Int. J. Comput. Vis.* **113** 54–66
[13] Schemmel J, Brüderle D, Grübl A, Hock M, Meier K and Millner S 2010 A wafer-scale neuromorphic hardware system for large-scale neural modeling *2010 IEEE Int. Symp. on Circuits and Systems (ISCAS)* (IEEE) pp 1947–50
[14] Benjamin B V *et al* 2014 Neurogrid: a mixed-analog–digital multichip system for large-scale neural simulations *Proc. IEEE* **102** 699–716
[15] Merolla P A *et al* 2014 A million spiking-neuron integrated circuit with a scalable communication network and interface *Science* **345** 668–73
[16] Furber S B, Galluppi F, Temple S and Plana L A 2014 The spinnaker project *Proc. IEEE* **102** 652–65
[17] Calimera A, Macii E and Poncino M 2013 The human brain project and neuromorphic computing *Funct. Neurol.* **28** 191
[18] Davies M *et al* 2018 Loihi: a neuromorphic manycore processor with on-chip learning *IEEE Micro* **38** 82–99
[19] Gallego G *et al* 2020 Event-based vision: a survey *IEEE Trans. Pattern Anal. Mach. Intell.* **44** 154–80
[20] Caviglia S, Pinna L, Valle M and Bartolozzi C 2016 An event-driven POSFET taxel for sustained and transient sensing *2016 IEEE Int. Symp. on Circuits and Systems (ISCAS)* pp 349–52
[21] Vitale A, Renner A, Nauer C, Scaramuzza D and Sandamirskaya Y 2021 Event-driven vision and control for UAVS on a neuromorphic chip *2021 IEEE Int. Conf. on Robotics and Automation (ICRA)* (IEEE) pp 103–9
[22] Dupeyroux J, Hagenaars J J, Paredes-Vallés F and de Croon G C 2021 Neuromorphic control for optic-flow-based landing of MAVS using the Loihi processor *ICRA 2021: IEEE Inte. Conf. on Robotics and Automation* (IEEE) pp 96–102
[23] Rueckert E, Kappel D, Tanneberg D, Pecevski D and Peters J 2016 Recurrent spiking networks solve planning tasks *Sci. Rep.* **6** 21142
[24] Weber D, Gühmann C and Seel T 2020 Neural networks versus conventional filters for inertial-sensor-based attitude estimation *2020 IEEE 23rd Int. Conf. on Information Fusion* (IEEE) pp 1–8
[25] Stagsted R, Vitale A, Binz J, Renner A, Larsen L B and Sandamirskaya Y 2020 Towards neuromorphic control: a spiking neural network based PID controller for UAV *Robot. Sci. Syst.* **6** 75
[26] Furrer F, Burri M, Achtelik M and Siegwart R 2016 Rotors—a modular gazebo MAV simulator framework *Stud. Comput. Intell.* **625** 595–625
[27] Pehle C and Pedersen J E 2021 Norse—a deep learning library for spiking neural networks https://norse.ai/docs/ (online). Available: https://doi.org/10.5281/zenodo.4422025
[28] Kingma D and Ba J 2014 Adam: a method for stochastic optimization *Int. Conf. on Learning Representations* vol 12
[29] Zhang M, Lucas J, Ba J and Hinton G E 2019 Lookahead optimizer: *k* steps forward, 1 step back *Advances in Neural Information Processing Systems* vol 32
[30] Neftci E O, Mostafa H and Zenke F 2019 Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks *IEEE Signal Process. Mag.* **36** 51–63
[31] Zenke F and Ganguli S 2018 SuperSpike: supervised learning in multilayer spiking neural networks *Neural Comput.* **30** 1514–41
[32] Zenke F and Vogels T P 2021 The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks *Neural Comput.* **33** 899–925
[33] Madgwick S O, Harrison A J and Vaidyanathan R 2011 Estimation of IMU and MARG orientation using a gradient descent algorithm *IEEE Int. Conf. on Rehabilitation Robotics* pp 179–85
[34] Mahony R, Hamel T and Pflimlin J-M 2008 Nonlinear complementary filters on the special orthogonal group *IEEE Trans. Autom. Control* **53** 1203–18
[35] Kalman R E 1960 A new approach to linear filtering and prediction problems *J. Basic Eng.* **82** 35–45
[36] Ribeiro M I 2004 *Kalman and Extended Kalman Filters: Concept, Derivation and Properties* vol 43 (Lisboa Portugal: Institute for Systems and Robotics) p 46
[37] Gui P, Tang L and Mukhopadhyay S 2015 Mems Based IMU for Tilting Measurement: Comparison of Complementary and Kalman Filter Based Data Fusion *IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)* pp 2004–9
[38] Shi Y and Eberhart R 1998 A modified particle swarm optimizer *1998 IEEE Int. Conf. on Evolutionary Computation Proc. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360)* (IEEE) pp 69–73
[39] Sabatini A M 2011 Kalman-filter-based orientation determination using inertial/magnetic sensors: observability analysis and performance evaluation *Sensors* **11** 9182–206
[40] Johansen T-A and Kristiansen R 2017 Quadrotor attitude estimation using adaptive fading multiplicative EKF *2017 American Control Conference (ACC)* (IEEE) pp 1227–32