2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Neuromorphic force-control in an industrial task: validating energy and latency benefits

Camilo Amaya[a], Evan Eames[a], Gintautas Palinauskas[b], Alexander Perzylo[a], Yulia Sandamirskaya[c], Axel von Arnim[a]

***Abstract*— As robots become smarter and more ubiquitous, optimizing the power consumption of intelligent compute becomes imperative towards ensuring the sustainability of technological advancements. Neuromorphic computing hardware makes use of biologically inspired neural architectures to achieve energy and latency improvements compared to conventional von Neumann computing architecture. Applying these benefits to robots has been demonstrated in several works in the field of neurorobotics, typically on relatively simple control tasks. Here, we introduce an example of neuromorphic computing applied to the real-world industrial task of object insertion. We trained a spiking neural network (SNN) to perform force-torque feedback control using a reinforcement learning approach in simulation. We then ported the SNN to the Intel neuromorphic research chip Loihi interfaced with a KUKA robotic arm. At inference time we show latency competitive with current CPU/GPU architectures, and one order of magnitude less energy usage in comparison to traditional low-energy edge-hardware. We offer this example as a proof of concept implementation of a neuromoprhic controller in real-world robotic setting, highlighting the benefits of neuromorphic hardware for the development of intelligent controllers for robots.**

I. INTRODUCTION

Moving towards the integration of intelligent robots in daily life, sustainability considerations require stricter optimization of energy consumption not only for actuation, but also for the computing involved in robot control. The current energy requirements of GPUs and CPUs either limit the scale and "intelligence" of edge computation, or require high-latency communication protocols for cloud computing. While moving an AI system such as IBM Watson or ChatGPT to the edge requires killowatts of power for continuous inference[1] [1], [2], the human brain accomplishes much more complex behaviour at a tiny fraction of this [3]. Biological inspiration therefore continues to influence both algorithm and hardware development in the field of neuromorphic computing, with an increasing number of applications in robotics.

Neuromorphic hardware refers to a novel hardware architecture that uses principles of computing in biological brains and neural systems and has been shown to drastically improve latency and energy usage for many computing tasks [4], in particular ones that rely on recurrent, temporal, and sparse computation, as often the case in motion planning and control [5]. Typically, following biological inspiration, neuromorphic processors realise in hardware so called Spiking Neural Networks (SNNs) [6], [7]. These differ from more ubiquitous Artificial Neural Networks (ANNs) in better exploitation of sparsity and recurrency through asynchronous multi-core processing (there is no global clock) and statefull neurons. Moreover, abundance of local memory enables efficient continual learning, along with the neural state updates. In recent years there has been an increasing interest in using neuromorphic hardware and SNNs for their advantages in energy and time efficiency in robotics — a domain referred to as "neurorobotics". We have proposed a roadmap for future development and deeper inspiration from biology for such systems [8].

Although a young area of research, neurorobotics has begun to see some first results. Examples include automotive object avoidance [9], SLAM [10], underwater propulsion [11], drone control [12], [13], and grasping force control [14]. See [15] for an overview. On the simulation front, the Neurorobotics Platform has been developed in the European Human Brain Project specifically as a sandbox for such applications [16].

However, as of yet the bulk of research has been carried out either partially or entirely within simulation [11], [17], [18], [19], [20]; with neuromorphic sensors paired with classical non-spiking hardware [21], [22], [23]; or with simple motion profiles (e.g. 1-dimensional movement) that do not constitute full use cases [24], [25], [26], [12], [14], [13]. Other efforts have used customized neuromorphic circuits in which direct energy/latency measurements are not possible [27] or hybrid approaches that are only partially guided by neuromorphic hardware [28]. See [29] for a full review. The consequence is that expected neuromorphic benefits cannot be inferred for real-world systems with practical applications.

To address this, we present a real-world neurorobotic system in which a robotic arm equipped with a force-torque (FT) sensor accomplishes an insertion task while fully controlled by neuromorphic hardware. The classic robotic insertion task has been chosen as it serves as the foundation for a number of related sub-tasks such as screw insertion, cable attachment, part assembly, etc. Additionally, the "peg-in-hole" task has historically served as a benchmark for robotic integration of new algorithms, sensors, and hardware [30], [31], [32], [33], [34], [35], [36], etc (see [37] for an overview).

[a] Department of Neuromorphic Computing, fortiss – Research Institute of the Free State of Bavaria, Munich, Germany <LastName>@fortiss.org

[b] The author was with the Department of Neuromorphic Computing, fortiss. He is now with Robominds GmbH, Munich, Germany gpa@robominds.de

[c] ZHAW Zurich University of Applied Sciences/Intel Neuromorphic Computing Lab yulia.sandamirskaya@zhaw.ch

[1] Ignoring training energy costs.

The arm is trained in simulation using spiking reinforcement learning (RL). The trained network is then ported to an Intel Loihi neuromorphic research chip, which is connected to our KUKA robotic arm. We use sim2real techniques to accomplish insertion with the real robot. An accompanying video can be viewed at https://tinyurl.com/y2szkbyb.

This RL in simulation + sim2real technique is similar to that recently applied to drone and quadrupedal maneuvering using conventional ANNs [38], [39], [40], [41]. Our approach is similar, but realised with SNNs and deployed on a neuromorphic chip. We argue that, as the hardware matures, this could lead to energy efficient trained controllers for a broad deployment of intelligent robots. Our experiments have shown one order of magnitude less energy usage in comparison to current SotA non-neuromorphic edge-optimized hardware at a similar latency.

We highlight the following novel contributions of this work: this is the first real-world non-trivial robotic use-case (peg-in-hole insertion with an industrial robotic arm) fully guided by neuromorphic hardware. In this setting, energy and latency measurements of neuromorphic hardware were herein performed for the first time. We have found energy spent on running the network on the neuromorphic chip to be in the microjoule range. Moreover, this is the first neurorobotic system to incorporate a force-torque sensor.

II. Methodology

We first develop a simulation of the robotics setup, train the neural network based controller with spiking-RL using Pytorch, port the network to neuromorphic hardware, and finally run the trained policy on the real robot, applying various techniques to address the Sim2Real gap. A visualization of these steps is shown in Figure 1.

A. Simulation Setup

The setup is simulated within the Neurorobotics Platform (NRP) [16], developed as part of the Human Brain Project [42] and based on ROS [43] and Gazebo [44]. A robotic arm with 7 DOF mounted to a table is modelled. The end-effector consists of a cylindrical peg with a flat end and a 6-axis FT sensor. The table consists of a board with a circular hole. The peg can fit in the hole with a clearance of $\approx$ 1 mm, and is initialized as per a Gaussian distribution centered on the hole with $\sigma = 2$ cm. This setup closely resembles the one used by previous teams (i.e. [34]). The setup is shown in Figure 2. The full details of the simulation setup and underlying programming have been outlined in a separate complimentary publication [45]. It primarily explores the implementation of RL within the NRP.

B. Reinforcement Learning

The simulated robotic arm is trained to find the hole using the spiking reinforcement learning technique described in [46]: a Population-encoded Spiking Actor Critic method (PopSAN). We base our state/action convention on that previously used by other teams in non-SNN works [37]. The state space consists of $[\vec{x}, \vec{\theta}, \vec{F}, \vec{\tau}]$: position, angle, forces and torques (relative to the end-effector). These are encoded into spikes using population encoding, in which neurons represent discrete real numbers. A continuous input can be encoded through multiple neurons spiking in various combinations. To arrive at this, in our case the spiking actor network was tuned such that each of the 13 dimensions of the state space were encoded by 10 neurons (for a total of 130). The policy returns an action consisting of $[\vec{x}, \vec{\theta}]$: a new target cartesian position and angle for the end-effector (returned as spikes, and then decoded into 6 real values for the low-level controller[2]). Each of the values in the action space is similarly represented by 10 neurons.

The network connected to the population-encoded neurons was a fully connected SNN composed of *Leaky Integrate & Fire* (LIF) neurons arranged into layer sizes: $130 \times 256 \times 256 \times 60$, and was implemented using Intel's NxSDK software library [5]. The critic networks are fully connected ANNs with ReLU activation functions for the hidden layers and hyperbolic tangent activation functions for the output layer. Both Q-networks had sizes $19 \times 256 \times 256 \times 1$. For further details regarding the networks and parameters used refer to [46].

After experimenting with a number of reward functions for RL, we settled on the following dense engineered reward:

$$R = w_1||f_d - f||_2 + w_2||\tau_d - \tau||_2 + w_3||z_d - z||_2 \quad (1)$$

Here f_d, τ_d, z_d are the desired force, torque, and depth. Similarly f, τ, z are the measured values for these three quantities. The weights w allow us to adjust the importance we assign to each of these terms. Recall that the goal is to learn insertion based on the force-torque profile experienced by the end-effector. This is the motivation behind the first two terms. The third term serves to quickly teach the arm to remain on the table. Without this term the number of training episodes becomes impractically large.

For our purposes, we take f_d and τ_d to be zero (this encourages a gentle insertion with minimal forces and torques). z_d is set to be –0.07 m — the bottom z coordinate of the hole. The w terms are heuristically adjusted based on the resulting behaviour. Explicitly, assigning high weight to the force and torque terms (effectively penalties, with $f_d = \tau_d = 0$) makes the arm hesitant to touch the table. Conversely, assigning these terms low weight can lead to erratic behaviour and dangerous forces that could damage the real set-up. For the z term, high weight leads to less exploration in favour of simply pushing down, and low weight leads to indifference to the actual insertion. We set weights $\vec{w} = [0.05, 0.05, 0.9]$.

C. Neuromorphic-Hardware-in-the-Loop

The policy is initially trained on an SNN simulated in PyTorch. Simultaneously, SNNs with the same architecture are defined in the corresponding frameworks for different neuromorphic hardware platforms (NxSDK for Loihi 1 and

[2] Notice that even though the robot has 7 DOF, the policy does not control the nullspace as it is not deemed relevant to the task. The nullspace could be used in future works to optimize other metrics, such as energy efficient joint control, but this was out of the scope of this project.

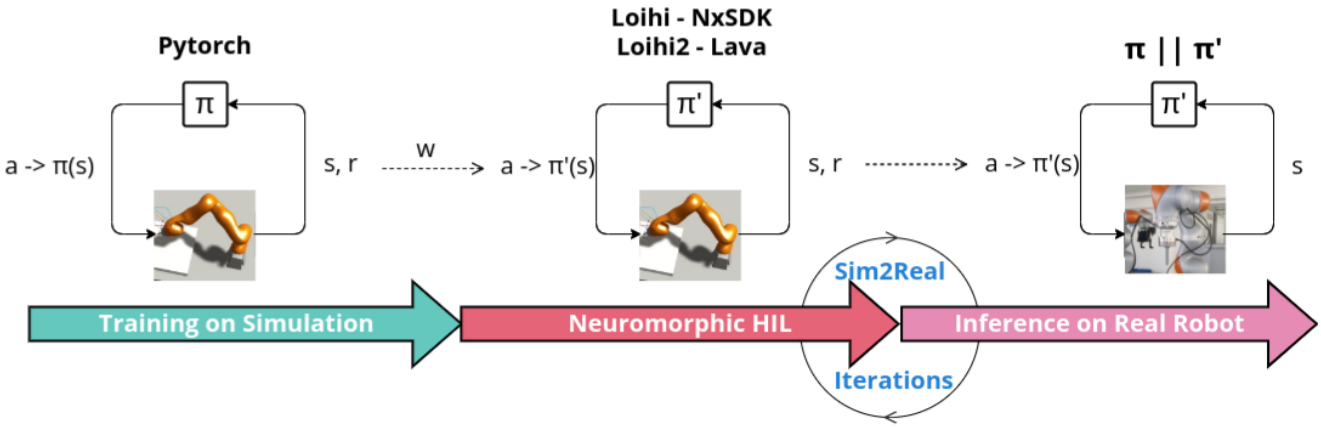

Fig. 1: Learning and inference approach. HIL refers to 'Hardware-In-Loop". An in-depth look at the third step can be found in Figure 4.

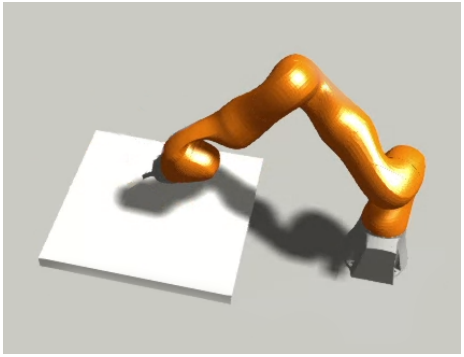

Fig. 2: Neurorobotics simulation setup.

LAVA[3] for Loihi 2 [47]). The trained weights can then be loaded onto the equivalent networks. The models may differ, as Loihi uses 9-bit precision values to describe the synaptic weights and neuron states, while the simulation allows for full precision values (64-bit floats). However, the differences were found to be negligible for this application. More specifically, we used two Intel Loihi 1 research chips in a USB-stick form factor system called Kapoho Bay [48] (seen in Figure 3), and a single Loihi 2 chip (Oheogulch) accessed remotely through the Intel Neuromorphic Research Cloud.

D. Real Robot Setup

We use a KUKA IIWA arm with 7 DOF[4]. As in the simulation the end-effector consists of a solenoidal peg attached to a 6-axis FT sensor. The box containing the hole is 3D printed, as is the end-effector peg. An image of the experimental setup is shown in Figure 3.

The spiking neural network trained in Section II-B controls the robot by providing a series of target poses. These targets can be connected through a smoothed path for reducing jerk in the motion, finally the path can be fed to a low-level impedance controller. The connection between these components is depicted in Figure 4.

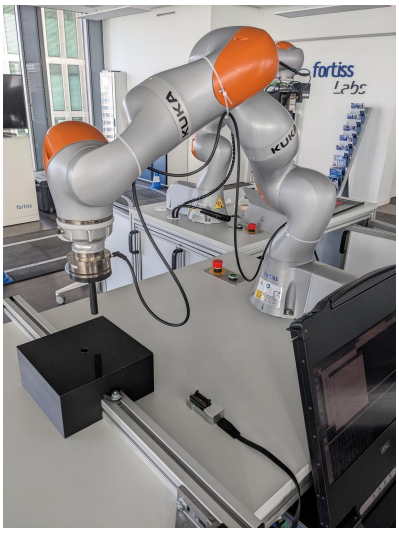

Fig. 3: Demonstrator setup with a KUKA IIWA 7 R800 robot and attached cylindrical peg, black target box with hole, and Kapoho Bay containing two Loihi chips (on the tabletop).

1) Controller: A low level compliant controller was designed and implemented both for simulation as well as for the real robot. More precisely, a Cartesian Impedance Controller (CIC) was implemented and tuned using the methodology outlined in [49]. The controller was implemented with the KUKA Fast Research Interface (FRI).

2) Path Updating: Small changes in the force-torque and positional values being measured from one moment to the next can trigger the policy to output a vastly different action from the previous one. Thus, the path must constantly be

[3]https://github.com/lava-nc
[4]See footnote 2

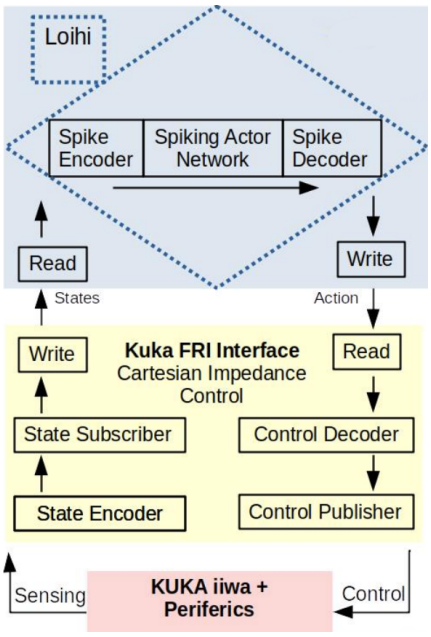

Fig. 4: An overview of the communication mechanism between the high-level neuromorphic controller running on Loihi and the low level controller operating on the FRI.

updated as new actions are received. For the compliant controller to accommodate smooth transitions between paths we use a fifth-degree polynomial for our path interpolation. This is chosen as it has been shown to reduce jerk, which, in turn, reduces manipulator wear and improves the trajectory accuracy and speed [50].

With the receipt of a new action from Loihi, the path is recalculated using the current pose, current velocity, current acceleration, and the final position (specified by the action). The final velocities and accelerations are taken to be zero.

3) Sim2Real: To bridge the Sim2Real gap we employ both Domain Randomization and System Identification. These methods have been shown to work well for robotic manipulation tasks when first training in simulation [51]. As the real coefficient of surface friction cannot be perfectly known (and can vary if switching end-effectors or surfaces) we allow the coefficient to take three values within the simulation $[0.34, 0.38, 0.42]$. From these, one is picked at the initialization of each training episode. These values are based upon repeated measurements of the real coefficient of friction on the printed surface. Multiple values assure robustness to variations in real surface friction [52].

Additionally, the sensor force-torque noise distribution was characterized and employed during training. The noise profile follows a Gaussian distribution and is modelled on the slight changes in these values felt by the real FT sensor when not touching any surface.

Finally, we added a scaling factor to the received target orientations such that the amplitude of angular movements matches those in simulation. The mismatch was mainly due to unmodeled differences in the controller with respect to chattering between consecutive non-adjacent orientations. This could be acknowledge either by making the simulated controller more closely resemble the real equivalent, including safety measurements and interpolators to prevent actuator damage, or by adding a component in the reward function during training to prevent said chattering behaviour from being learned. However, for the sake of the task at hand we found that a simple scaling factor in the orientation provide a simpler solution.

III. Results

A. Training

The training (Figure 1, left) was run for 100 epochs, with 500 episodes and 2000 interactions each. Figure 5 shows the training profile of 20 different random seeds in terms of mean return value and corresponding standard deviation. We find a 100% insertion success rate using a trained policy (across 50 runs, where the system is deemed to have failed if it does not insert within 30 s). Furthermore, there was no detected performance drop after porting the policy to real neuromorphic hardware while still using the simulated robotic arm (Figure 1, center). Even though the models differ slightly due to quantization effects, these differences represent only negligible behavioural variations and still lead to the same success rates.

Fig. 5: Learning curves showing the results of training with 20 random seeds, the mean performance and the standard deviation.

B. Real-World Performance

When the policy is initially run on the real robot (Figure 1, right), we find an insertion success rate of 0% (across 10 runs, where the system is deemed to have failed if it does not insert within 30 s). After the Sim2Real techniques (Section II-D.3) we find a 100% insertion success rate across 50 runs. Using success rate to compare the simulated and real implementations is therefore not possible, so we instead use the time to insertion (shown in Table I). All values are calculated across 50 runs.

A similar median and minimum insertion time is observed. Note that the mean real insertion time is roughly double

the simulated time. This is due to a small number of outlier runs in which the peg has trouble navigating the complex real friction profile and temporarily gets stuck (which also explains the maximum insertion time discrepancy). Yet, perfect friction modelling is not our goal, and median insertion time on the order of a few seconds is comparable with SotA implementations (see, for example, Figure 16 in [36]). We now move to the energy and latency.

TABLE I: Time to Insertion

	Simulation	Real
Mean	3.7 s	8.4 s
Median	3.4 s	5.3 s
Minimum	2.5 s	2.2 s
Maximum	7.4 s	28.1 s

C. Latency & Energy

Measuring the latency on the Loihi chip using the state probes we find the time profile shown in Figure 6. We find slightly smaller latency on the Loihi 2 chip (1.5 ± 0.10 ms).

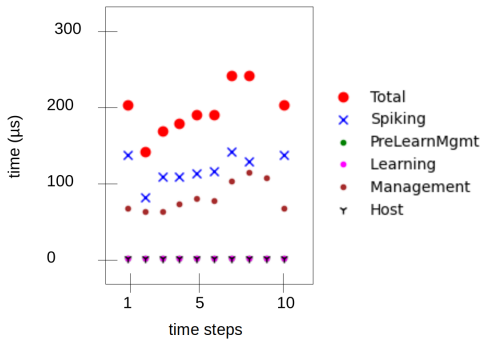

Fig. 6: SNN execution time on Loihi 1 measured with state probes over 9 steps — equivalent to one inference. The total is 1.8 ms (Table II). Pre-learning management and learning are zero as the learning was off-line (and is additionally not relevant for inference).

We find a per inference dynamic energy cost (removing the background energy cost of running the hardware) of 52 ± 17 µJ. This corresponds to a few tens of milliwatts. These results, in addition to comparisons with CPU and GPU values, are summarized in Table II.

TABLE II: Energy and Time Profiling

Hardware	E_{dynamic} [µJ]	Latency [ms]
CPU †	3800	1.4 ± 0.1
GPU	$\sim$ a few 100‡	2.0 ± 0.1*
Loihi 1**	—	1.8
Loihi 2	53 ± 17	1.5 ± 0.1

† CPU: 11th Gen Intel® Core™ i7-1165G7 @ 2.80GHz × 8.

‡ NVIDIA® / Pascal 256 CUDA cores @ 1300 MHz / Jetson TX2 ([53] Alg. 1 + personal communication with author)

* NVIDIA® / Mesa Intel® Xe Graphics (TGL GT2) / GeForce MX550

** Note: Energy values are not available on Loihi 1 due to probe limitations.

IV. Discussion

A. Latency & Energy

Contrasting the recorded Loihi energy values with SotA non-neuromorphic edge computational hardware (Table II), the potential benefits are evident. Currently, optimized edge-hardware requires energy on the order of hundreds of µJ per inference for similar tasks/networks [53], [54], [55], an order of magnitude more than neuromorphic hardware. Such hardware could therefore allow for more complex computations on autonomous robots, as well as orders-of-magnitude longer times between recharging. Additionally, non-neuromorphic edge-hardware often achieves low power consumption at the cost of latency (generally on account of cloud communication). For instance, [54] cite latencies on the order of hundreds of milliseconds, compared to a few milliseconds by computing directly on Loihi (Figure 6). It is also important to remember that Loihi is intended as a flexible research platform and has not been optimized for specific tasks.

Previous RL implementations of peg-in-hole (e.g. [34], [56], [31]) generally invoke ANNs of a similar size to the SNN introduced here. Therefore, we should expect that the inference-time energy improvements over CPU and GPU (Table II) apply here. Of course, inference (and computation in general) represents only one part of the entire robotic control system. For larger robots (10s of kg), the power requirements of the mechanical actuators are roughly one order of magnitude larger than the power requirements of computation [57]. Yet for sub-kilogram robots computation is likely to be a non-negligible source of power-consumption. As edge-robots become smaller and the tasks we expect them to perform become more complex, efficient compute will become even more imperative.

The measured energy and latency results are comparable to those reported in simulated neuromorphic use cases, hence validating the idea that the benefits of neurorobotic systems continue to hold when moved from simulations and simplistic use cases to more complex real-world applications. For example, [19] reports $\approx$ 200 µJ and $\approx$ 1 ms per inference using a slightly larger network on Loihi 1 for simulated robotic control. [12] run a 4-layer SNN on Loihi 1D drone control. Although energy values are not reported, they do find a latency of 0.05 ms per step, corresponding to an equivalent of $\approx$ 0.5 ms per inference. This value may be

smaller than what we find on account of especially sparse data (coming from a neuromorphic camera). We noted that the latency values for CPU and GPU vary largely between devices. Indeed, [19] cite 15 ms latency on CPU, 10 times larger than our value. Ultimately we find similar latency between CPU and neuromorphic hardware.

It should be noted that Table II refers to the computation times on-chip. On account of a known I/O bottleneck in Loihi 1 the robot-chip communication, which in theory should be negligible, ultimately increases the latency to ≈ 16 ms. Although addressed in Loihi 2 [58], we do not have an on-site physical chip, and therefore cannot measure the end-to-end latency on account of the cloud-communication cost. There is no reason, however, that the end-to-end latencies should be larger than what is quoted in Table II. We also remind the reader that the energy per inference represents only one part of the overall energy required for computation. Loihi currently requires significant power for overhead operations (such as for spike I/O - handled by a separate FPGA). These are not included, as they are expected to be optimized with subsequent chip releases, to the point that the inference energy is primary (as was the case for classical chips).

B. Application Specific / Peg-in-hole

The insertion times (Table I) are comparable between simulation and the real robot. We find quantization effects on Loihi (previously explored by [59]) to be negligible. We notice that, even when applying domain randomization in simulation training to vary the friction coefficient (Section II-D.3) the movement behaviour still differs somewhat due to the complex friction profile between the peg and surface. Notably, on the real surface, the peg will often initially not move, and then jerk forward when the friction is overcome. This jerking was not captured in simulation, although insertion on the real robot is still successful regardless. Velocity and acceleration limits are the standard KUKA preset values. We do not expect varying the limits would affect the outcome, although this would need to be explicitly tested.

We had initially hypothesized that the arm, using the FT sensor and ability to move in angular space, would learn to feel the force-torque profile corresponding to being on the edge of the hole. We find that the trained policy is such that the angular movement is quite small ($\pm \approx 1^\circ$ for all three axes). This means that generally the peg is held approximately vertical. Additionally, during the exploration phase, the trained policy returns actions sending the end-effector back-and-forth into the positive and negative x and y directions (relative to the hole). Although the peg appears to sometimes react to arriving at the edge of the hole (by moving towards the hole centre), in a number of runs we certainly see the peg slipping into the hole, likely by accident, from this learned sweeping behaviour. However, in the case where the peg only slips part-way into the hole, the slight angular movements do appear to help it arrive at the bottom.

In a future work, it would be worthwhile exploring a larger policy network allowing for more complex behaviour, more robust noise modelling (to account for the above-mentioned jerky motion), and holes at different angles to make the sweeping motion insufficient for insertion.

C. Neurorobotic Outlook

We have demonstrated robotic insertion using a neuromorphic high-level controller as a proof of concept, meant to emphasize that neuromorphic computing is no longer simply up-and-coming, but rather sufficiently mature to move towards tackling real robotic use cases. This implementation is additionally intended to act as a software base for future neurorobotic development.

Indeed, flexible research platforms such as Loihi can already be used as test-beds and prototyping tools for current non-research neuromorphic and mixed-signal devices with SoTA performance[5]. Neuromorphic hardware also allows for "online" and "continuous" learning unique to SNNs and inspired by the human brain which, although found not to be needed for successful insertion in our use case, we nonetheless hope to explore in a future publication.

Looking forward, it is worth mentioning that current actuators do not operate on spiking principles. With neuromorphic hardware and sensors already in existence, spike-based actuators would allow for neurorobots in which energy and latency is not lost to encoding and decoding spikes between neuromorphic and non-neuromorphic components. Additional research is here needed to unlock the true potential of neurorobots.

Ultimately, and as with the first RL peg-in-hole implementation, we hope that this first neuromorphic implementation contributes to an exciting new domain of robotics and automatization — that of neurorobotics.

CODE

Code available upon request.

ACKNOWLEDGEMENTS

The research at fortiss was supported by the HBP Neurobotics Platform funded from the European Union's Horizon 2020 Framework Program for Research and Innovation under the Specific Grant Agreements No. 945539 (Human Brain Project SGA3).

REFERENCES

[1] Larry Greenemeier. Will ibm's watson usher in a new era of cognitive computing? *Scientific American*, November 2013.

[2] Tom B. Brown *et al.* Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[3] Vijay Balasubramanian. Brain power. *Proceedings of the National Academy of Sciences*, 118(32):e2107022118, 2021.

[4] Dennis V. Christensen *et al.* 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2(2):022501, may 2022.

[5] Mike Davies *et al.* Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021.

[5]eg. DYNAP-SE, ROLLS, Innaterra, etc.

[6] 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.

[7] John V. Arthur and Kwabena A. Boahen. Silicon-neuron design: A dynamical systems approach. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58:1034–1043, 2011.

[8] Yulia Sandamirskaya, Mohsen Kaboli, Jorg Conradt, and Tansu Celikel. Neuromorphic computing hardware and neural architectures for robotics. *Science Robotics*, 7(67):eabl8419, 2022.

[9] J. Parker Mitchell, Grant Bruer, Mark E. Dean, James S. Plank, Garrett S. Rose, and Catherine D. Schuman. Neon: Neuromorphic control for autonomous robotic navigation. In *2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pages 136–142, 2017.

[10] Guangzhi Tang, Arpit Shah, and Konstantinos P. Michmizos. Spiking neural network on neuromorphic hardware for energy-efficient unidimensional SLAM. *CoRR*, abs/1903.02504, 2019.

[11] Emmanouil Angelidis *et al.* A spiking central pattern generator for the control of a simulated lamprey robot running on spinnaker and loihi neuromorphic boards. *Neuromorphic Computing and Engineering*, 1, 08 2021.

[12] Antonio Vitale, Alpha Renner, Celine Nauer, Davide Scaramuzza, and Yulia Sandamirskaya. Event-driven vision and control for uavs on a neuromorphic chip. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[13] F. Paredes-Vallés, Jesse J. Hagenaars, Julien Dupeyroux, Stein Stroobants, Ying Xu, and G.C.H.E. de Croon. Fully neuromorphic vision and control for autonomous drone flight. *ArXiv Preprint*, abs/2303.08778, 2023.

[14] Chao Bao, Tae-Ho Kim, Amirhossein Hassanpoor Kalhori, and Woo Soo Kim. A 3d-printed neuromorphic humanoid hand for grasping unknown objects. *iScience*, 25(4):104119, 2022.

[15] Chiara Bartolozzi, Giacomo Indiveri, and Elisa Donati. Embodied neuromorphic intelligence. *Nature Communications*, 13:1024, 02 2022.

[16] Egidio Falotico *et al.* Connecting artificial brains to robots in a comprehensive simulation framework: The neurorobotics platform. *Frontiers in Neurorobotics*, 11, 2017.

[17] Alex Volinski, Yuval Zaidel, Albert Shalumov, Travis DeWolf, Lazar Supic, and Elishai Ezra. Data-driven artificial and spiking neural networks for inverse kinematics in neurorobotics. *Patterns*, 3:100391, 11 2021.

[18] Marina González-Álvarez, Julien Dupeyroux, Federico Corradi, and Guido de Croon. Evolved neuromorphic radar-based altitude controller for an autonomous open-source blimp. *CoRR*, abs/2110.00646, 2021.

[19] Travis DeWolf, Kinjal Patel, Pawel Jaworski, Roxana Leontie, Joe Hays, and Chris Eliasmith. Neuromorphic control of a simulated 7-dof arm using loihi. *Neuromorphic Computing and Engineering*, 3(1):014007, feb 2023.

[20] Gintautas Palinauskas, Camilo Amaya, Evan Eames, Michael Neumeier, and Axel Von Arnim. Generating event-based datasets for robotic applications using mujoco-esim. In *Proceedings of the 2023 International Conference on Neuromorphic Systems*, ICONS '23, New York, NY, USA, 2023. Association for Computing Machinery.

[21] Rajkumar Muthusamy, Abdulla Ayyad, Mohamad Halwani, Dewald Swart, Dongming Gan, Lakmal Seneviratne, and Yahya Zweiri. Neuromorphic eye-in-hand visual servoing. *IEEE Access*, 9:55853–55870, 2021.

[22] Abdulla Ayyad, Mohamad Halwani, Dewald Swart, Rajkumar Muthusamy, Fahad Almaskari, and Yahya Zweiri. Neuromorphic vision based control for the precise positioning of robotic drilling systems. *Robot. Comput.-Integr. Manuf.*, 79(C), feb 2023.

[23] Wallace Lawson, Anthony Harrison, and J. Gregory Trafton. Sigma-delta networks for robot arm control. In *Proceedings of the 2023 Annual Neuro-Inspired Computational Elements Conference*, NICE '23, page 35–40, New York, NY, USA, 2023. Association for Computing Machinery.

[24] Rasmus Stagsted, Antonio Vitale, Jonas Binz, Alpha Renner, Leon Larsen, and Yulia Sandamirskaya. Towards neuromorphic control: A spiking neural network based pid controller for uav. In *Robotics: Science and Systems 2020*, 07 2020.

[25] Yuval Zaidel, Albert Shalumov, Alex Volinski, Lazar Supic, and Elishai Ezra Tsur. Neuromorphic nef-based inverse kinematics and pid control. *Frontiers in Neurorobotics*, 15, 2021.

[26] Alejandro Linares-Barranco, Fernando Perez-Peña, Angel Jiménez-Fernandez, and Elisabetta Chicca. Ed-biorob: A neuromorphic robotic arm with fpga-based infrastructure for bio-inspired spiking motor controllers. *Frontiers in Neurorobotics*, 14, 2020.

[27] Ashwin Sanjay Lele, Yan Fang, Justin Ting, and Arijit Raychowdhury. An end-to-end spiking neural network platform for edge robotics: From event-cameras to central pattern generation. *IEEE Transactions on Cognitive and Developmental Systems*, 14:1092–1103, 2022.

[28] Michael Ehrlich *et al.* Adaptive control of a wheelchair mounted robotic arm with neuromorphically integrated velocity readings and online-learning. *Frontiers in Neuroscience*, 16, 2022.

[29] Muhammad Aitsam, Sergio Davies, and Alessandro Di Nuovo. Neuromorphic computing for interactive robotics: A systematic review. *IEEE Access*, 10:122261–122279, 2022.

[30] H. Brussel and J. Simons. Adaptable compliance concept and its use for automatic assembly by active force feedback accommodations. *Proceedings of the 9th International Symposium on Industrial Robots*, pages 167–181, 01 1979.

[31] M. Nuttin and H. Brussel. Learning the peg-into-hole assembly operation with a connectionist reinforcement technique. *Computers in Industry*, 33:101–109, 08 1997.

[32] Kuangen Zhang, MinHui Shi, Jing Xu, Feng Liu, and Ken Chen. Force control for a rigid dual peg-in-hole assembly. *Assembly Automation*, 37:200–207, 04 2017.

[33] B.H. Yoshimi and P.K. Allen. Active, uncalibrated visual servoing. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 156–161 vol.1, 1994.

[34] Tadanobu Inoue, Giovanni De Magistris, Asim Munawar, Tsuyoshi Yokoya, and Ryuki Tachibana. Deep reinforcement learning for high precision assembly tasks, 2017.

[35] Cristian C. Beltran-Hernandez, Damien Petit, Ixchel G. Ramirez-Alpizar, and Kensuke Harada. Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach. *Applied Sciences*, 10(19):6923, oct 2020.

[36] Monica Sileo, Nicola Capece, Monica Gruosso, Michelangelo Nigro, Domenico D. Bloisi, Francesco Pierri, and Ugo Erra. Vision-enhanced peg-in-hole for automotive body parts using semantic image segmentation and object detection. *Engineering Applications of Artificial Intelligence*, 128:107486, 2024.

[37] Jing Xu, Zhimin Hou, Zhi Liu, and Hong Qiao. Compare contact model-based control and contact model-free learning: A survey of robotic peg-in-hole assembly strategies. *CoRR*, abs/1904.05240, 2019.

[38] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Deep drone acrobatics. *CoRR*, abs/2006.05768, 2020.

[39] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.

[40] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

[41] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, Aug 2023.

[42] Alois Knoll and Mark-Oliver Gewaltig. Neurorobotics : A strategic pillar of the human brain project. pages 35–39. Science/AAAS Custom Publishing Office, 2016.

[43] Morgan Quigley *et al.* Ros: an open-source robot operating system. In *2009 ICRA Workshop on Open Source Software*, volume 3, 01 2009.

[44] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004.

[45] Camilo Amaya and Axel von Arnim. Neurorobotic reinforcement learning for domains with parametrical uncertainty. *Frontiers in Neurorobotics*, 17, 2023.

[46] Guangzhi Tang, Neelesh Kumar, Raymond Yoo, and Konstantinos P. Michmizos. Deep reinforcement learning with population-coded spiking neural network for continuous control, 2020.

[47] Garrick Orchard *et al.* Efficient neuromorphic signal processing with loihi 2. In *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 254–259, 10 2021.

[48] Mike Davies *et al.* Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[49] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger. Cartesian impedance control of redundant robots: recent results with the dlr-light-weight-arms. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 3, pages 3704–3709 vol.3, 2003.

[50] S. Macfarlane and E.A. Croft. Jerk-bounded manipulator trajectory planning: design for real-time applications. *IEEE Transactions on Robotics and Automation*, 19(1):42–52, 2003.

[51] Leon Sievers, Johannes Pitz, and Berthold Bäuml. Learning purely tactile in-hand manipulation with a torque-controlled hand, 04 2022.

[52] Lilian Weng. Domain randomization for sim2real transfer. *lilian-weng.github.io*, 2019.

[53] Seyyidahmed Lahmer, Aria Khoshsirat, Michele Rossi, and Andrea Zanella. Energy consumption of neural networks on nvidia edge boards: an empirical model. In *2022 20th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. IEEE, September 2022.

[54] Jie Tang, Shaoshan Liu, Liangkai Liu, Bo Yu, and Weisong Shi. Lopecs: A low-power edge computing system for real-time autonomous driving services. *IEEE Access*, 8:30467–30479, 2020.

[55] Lucas Martin Wisniewski, Jean-Michel Bec, Guillaume Boguszewski, and Abdoulaye Gamatié. Hardware solutions for low-power smart edge computing. *Journal of Low Power Electronics and Applications*, 12(4), 2022.

[56] Jianlan Luo, Eugen Solowjow, Chengtao Wen, Juan Ojea, and Alice Agogino. Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects. pages 2062–2069, 10 2018.

[57] Navvab Kashiri *et al.* An overview on principles for energy efficient robot locomotion. *Frontiers in Robotics and AI*, 5, 2018.

[58] Intel. Intel advances neuromorphic with loihi 2, new lava software framework and new partners. *(available at https://www.intel.com/content/www/us/en/newsroom/news/intel-unveils-neuromorphic-loihi-2-lava-software.html)*, 2022.

[59] Mahmoud Akl, Yulia Sandamirskaya, Florian Walter, and Alois Knoll. Porting deep spiking q-networks to neuromorphic chip loihi. In *International Conference on Neuromorphic Systems 2021*, ICONS 2021, New York, NY, USA, 2021. Association for Computing Machinery.