

Neuromorphic Computing for Autonomous Racing

Robert M. Patton, Catherine D. Schuman, Shruti R. Kulkarni, Maryam Parsa, J. Parker Mitchell

Computer Science and Math Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

N. Quentin Haas, Christopher Stahl, Spencer Paulissen, Prasanna Date, Thomas E. Potok

Computer Science and Math Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

Shay Snyder

East Tennessee State University
Johnson City, Tennessee, USA

ABSTRACT

Neuromorphic computing has many opportunities in future autonomous systems, especially those that will operate at the edge. However, there are relatively few demonstrations of neuromorphic implementations on real-world applications, partly because of the lack of availability of neuromorphic hardware and software, but also because of the lack of availability of an accessible demonstration platform. In this work, we propose utilizing the F1Tenth platform as an evaluation task for neuromorphic computing. F1Tenth is a competition wherein one tenth scale cars compete in an autonomous racing task; there are significant open source resources in both software and hardware for realizing this task. We present a workflow with neuromorphic hardware, software, and training that can be used to develop a spiking neural network for neuromorphic hardware deployment to perform autonomous racing. We present initial results on utilizing this approach for this small-scale, real-world autonomous vehicle task.

CCS CONCEPTS

• **Hardware** → **Neural systems**; • **Computing methodologies** → *Supervised learning*; *Bio-inspired approaches*.

KEYWORDS

neuromorphic computing, autonomous driving, evolutionary algorithms

ACM Reference Format:

Robert M. Patton, Catherine D. Schuman, Shruti R. Kulkarni, Maryam Parsa, J. Parker Mitchell, N. Quentin Haas, Christopher Stahl, Spencer Paulissen, Prasanna Date, Thomas E. Potok, and Shay Snyder. 2021. Neuromorphic

Notice: This manuscript has been authored in part by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (<http://energy.gov/downloads/doe-public-access-plan>).

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the United States government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

ICONS 2021, July 27–29, 2021, Knoxville, TN, USA

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-8691-3/21/07...\$15.00

<https://doi.org/10.1145/3477145.3477170>

Computing for Autonomous Racing. In *International Conference on Neuromorphic Systems 2021 (ICONS 2021)*, July 27–29, 2021, Knoxville, TN, USA. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3477145.3477170>

1 INTRODUCTION

With the rise of successful uses of artificial intelligence (AI) and machine learning (ML) for a wide variety of applications in the last decade, there has been a corresponding increase in interest in applying AI and ML methods to develop intelligent autonomous systems [11]. Many of these applications have hard constraints on physical deployments of AI and ML for autonomy, such as energy or power constraints and/or real-time processing constraints. To meet those challenges, custom hardware is likely to be required. Neuromorphic computing systems offer one path forward for resource-constrained intelligent autonomous systems. Neuromorphic computers are brain-inspired in their hardware implementation. They natively perform neural network-style computation and can do so typically with very low power [24].

There are several challenges associated with training and deploying a neuromorphic system within an autonomous system, especially for automatic control within the system. The first is the availability of neuromorphic hardware. Though large-scale neuromorphic hardware systems such as Intel's Loihi [8] have been developed, there are relatively few physical hardware systems available, and most of the research community is only able to access them in the cloud. This makes a physical demonstration in the real-world significantly more difficult, especially if real-time performance is of concern. The second key challenge is the availability of software and software simulations for neuromorphic implementations that enables rapid development, especially for control-oriented tasks. Most neuromorphic algorithms are targeted toward classification and prediction tasks [24]. Additionally, many real-world control tasks require domain expertise to develop demonstrations and thus often require partnerships and collaborations beyond the neuromorphic community. This limits the ability of any one neuromorphic researcher to develop their own demonstration. Because of these challenges, there have been relatively few real-world demonstrations of neuromorphic hardware on control tasks.

Autonomous vehicles are an intriguing use case for neuromorphic systems. They require real-time decision making, but they also require low size, weight, and power solutions. Additionally, the need for machine learning (ML) in autonomous driving is clear. We cannot enumerate every possible scenario that will be encountered in the real-world through a rule-based approach; instead, autonomous vehicles need ML approaches that can learn during training and

apply what has been learned in the real world to situations that were not encountered over the course of training.

As a first step on the path to neuromorphic computers for autonomous vehicles, in this work, we present a small-scale autonomous driving task based on the F1Tenth competition.¹ The F1Tenth competition is a 1/10th scale version of Formula One racing. To lower the barrier of entry, the F1Tenth community has developed the F1Tenth Autonomous Vehicle System, an open-source platform meant to be used for research and education. Additionally, simulated environments are provided for preliminary evaluation.

In this work, we highlight the F1Tenth platform as a demonstration use case for neuromorphic systems. We present a workflow for training spiking neural networks for neuromorphic deployment using a hardware accurate neuromorphic simulator, and we apply this workflow to the F1Tenth task in simulation. We present preliminary results on this task in simulation, and we discuss our plan for deployment onto the physical F1Tenth platform.

2 RELATED WORK

Neuromorphic computing is Turing-complete [7] and has been used to address problems in many areas such as graph theory [13, 15], epidemiological simulations [12] and anomaly detection [6]. Neuromorphic systems have also been used in autonomous robots. Hwu et al. leverage IBM TrueNorth to train a self-driving robot capable of traversing steep mountain paths by manually constructing a data set of path following behavior [14]. Chen et al. demonstrate NeuroSnake, which is a bio-inspired snake-like robot equipped with silico-retina sensor and SpiNNaker [3]. Apart from these, researchers have proposed toolboxes and frameworks for neuromorphic robotics [5, 10], as well as neuromorphic vision sensors for use in robotics [2, 4, 20].

We have previously utilized a similar workflow as the one proposed in this work with autonomous robotics tasks [1, 16]. In both of those cases, custom robots were built the task was to navigate an unfamiliar environment while avoiding obstacles. Both of those approaches used FPGA-based neuromorphic implementations, DANNA [9] and DANNA2 [17]. We also employed a similar neuromorphic workflow to deploy a neuromorphic solution for real-time engine control [25], in collaboration with transportation researchers. In all of these cases, we performed training in simulation (with simulated robots or a simulated engine) and then physically deployed the trained network to neuromorphic hardware that was then run in closed-loop with the actual robot or an actual internal combustion engine.

3 NEUROMORPHIC WORKFLOW

To enable neuromorphic deployment to a physical demonstration, several key components are required: hardware, software, and training. In this section, we briefly overview the components of the neuromorphic workflow used in the F1Tenth application.

3.1 Hardware: μ Caspian

We use μ Caspian [19] as our neuromorphic hardware implementation. μ Caspian is part of the Caspian neuromorphic development platform [18], which includes system software, hardware-accurate

software simulations, and hardware implementations on field programmable gate arrays (FPGAs). μ Caspian is a specific hardware instantiation of Caspian that is targeted towards low cost, low power edge deployment of neuromorphic systems. The μ Caspian development board includes a Lattice iCE40 UP5K1 FPGA on which the neuromorphic architecture is implemented. It also includes USB communication as well as on-board power monitoring capabilities.

In this work, we focus on training and testing in simulation; with that, we utilize μ Caspian’s hardware accurate software simulator, written in C++. We have previously demonstrated that we can seamlessly deploy a spiking neural network (SNN) trained with this hardware-accurate simulator to the physical μ Caspian board and the SNN behaves identically in hardware, as expected. The USB interface for the μ Caspian development board is included for ease of usability and deployment, but consumes a significant portion of the development board’s power. With the USB interface, the μ Caspian development board uses approximately 500 mW; without, the FPGA alone can operate at approximately 10-20 mW. FPGA-based neuromorphic systems are an excellent platform to perform development at low cost but to also provide some energy efficiency benefits as well. The μ Caspian system can accommodate SNNs of up to 256 neurons and 4096 synapses.

3.2 Software: TENNLab Framework

To interface with the underlying neuromorphic hardware and its own custom software system, we use the TENNLab Neuromorphic Software Framework [21]. The TENNLab framework provides a common high-level interface to multiple neuromorphic hardware systems and simulators. With this common software interface, algorithms and applications can be developed independently of the underlying neuromorphic hardware platform. The TENNLab framework provides an API that includes options to load an SNN onto the hardware, apply input spikes, read output spikes, and run the underlying simulation or hardware for some number of timesteps (among others). Additionally, the TENNLab framework provides spike encoding and decoding approaches, so that the application developer does not have to directly convert application state information to and from spikes. The framework supports a variety of spike encoding approaches, including population coding/binning and rate coding [23].

3.3 Algorithm: EONS

For training and designing an SNN for this task, we use Evolutionary Optimization for Neuromorphic Systems (EONS) [22]. EONS is based on evolutionary algorithms and trains SNNs for deployment to neuromorphic hardware. It designs all aspects of the SNN, including the number of neurons, number of synapses, and the connectivity between them, as well as all of the parameters of the network (synaptic delays, synaptic weights, and neuronal thresholds). EONS begins with randomly initialized potential SNNs to form a population and then performs an iterative process of SNN evaluation, selection, and reproduction. Because EONS relies entirely on a fitness evaluation score to drive the optimization, it is extremely flexible to apply to different types of applications, including classification, control, and anomaly detection. To use EONS

¹More information about the F1Tenth competition is available at <https://f1tenth.org/>

to train a network for a task, one must define this fitness evaluation score such that higher scores correspond to better performing networks. However, developing an appropriate fitness score is often more art than science; the fitness evaluation for the F1Tenth application is described in detail in Section 4.

4 F1TENTH APPLICATION

In this work, we focus on training in simulation, and we plan to deploy to a physical F1Tenth platform with μ Caspian as the on-board controller. In this work, we utilize the F1Tenth simulator, which is available as a Gym environment.² The F1Tenth gym environment allows the user to specify a track on which to train. By default, it uses a pre-defined test track, but there are also 23 1/10th scale Formula One tracks that are available for use. Here, we use five of the real-world tracks for training and another 20 for testing.

The simulator provides a variety of state information about the vehicle on the track. We use the LIDAR sensor as the observation, and we utilize information about distance traveled, collisions, and laps completed as part of our fitness evaluation. The LIDAR sensor provides 1080 individual beam values, sampled from a 270 degree forward-facing view of the car. We down-sample to 10 equally spaced LIDAR beams in this field of view, and those values are given as input to the SNN. The actions that can be applied to the simulator are speed and steering angle every 5 milliseconds. As such, the SNN will receive 10 LIDAR beams as input and needs to produce a speed and steering angle as output in less than 5 milliseconds to operate in real time. We use a binning encoding scheme with two bins and up to 8 spikes per bin. This encoding scheme was chosen because it has previously performed well on autonomous robot navigation tasks [23]. The SNN can select between three speed values (2, 5, 10) and 13 possible angle values (0, -0.01, 0.01, -0.03, 0.03, -0.05, 0.05, -0.07, 0.07, -0.1, 0.1, -0.3, 0.3). Whichever neuron spikes the most in the set of speed neurons (steering angle neurons) is selected as the output speed (steering angle).

The F1Tenth simulator has two stopping conditions: the car crashes into a wall (collision) or the car completes two laps. There are several behaviors that we want to encourage, besides completing two laps without colliding with a wall:

- Driving mostly straight, penalizing non-zero steering angles.
- Performing any driving maneuvers that substantially increases the distance traveled over what would be expected for the track (e.g., U-turns on the track).
- Complete the two laps as quickly as possible

As such, we define our fitness function as follows, where net is the network to be evaluated and tr is the track:

$$f(net, tr) = \begin{cases} \frac{d}{D_{tr}} & L < 2 \\ -1 & L = 2 \text{ \& } d - D_{track} > 10 \\ \frac{d}{D_{tr}} + \frac{1}{\sum_{i=0}^T \alpha_i} + \frac{10}{T} & \text{otherwise} \end{cases} \quad (1)$$

where d is the total distance traveled, D_{tr} is the nominal length to travel two laps for that track, α_i is the steering angle for time step i , and T is the total number of successful time steps completed

²More resources from the F1Tenth community, including the Gym environment, are available here: <https://github.com/f1tenth>

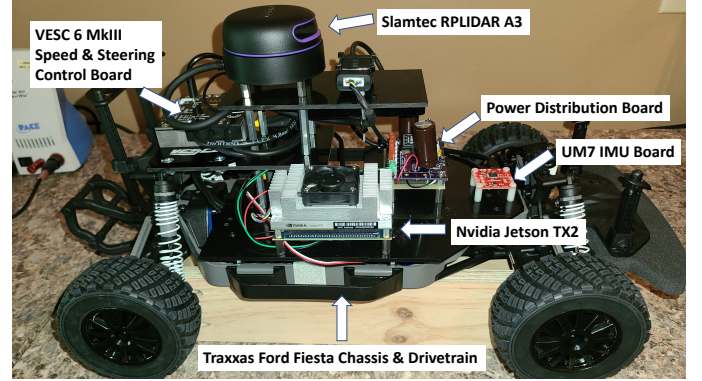


Figure 1: F1Tenth car with on-board hardware. In this work, we will replace the Nvidia Jetson TX2 with the μ Caspian implementation.

without colliding, and L is the total laps completed. The goal of this fitness function is to encourage covering as much distance as possible until two laps are completed. Once the networks can complete two laps for that track, if the distance is substantially more than the nominal distance, it is penalized significantly by setting the fitness score to -1. If two laps are completed and the distance is not substantially more, the fitness function then rewards smaller steering angles used throughout $\left(\frac{1}{\sum_{i=0}^T \alpha_i}\right)$ and faster speeds $\left(\frac{10}{T}\right)$.

This fitness evaluation was determined experimentally; in the future, we plan to explore different fitness evaluation approaches to encourage different behaviors. The full fitness evaluation as the average across five training tracks.

4.1 Physical Vehicle

The physical vehicle is shown in Figure 1. Based on the requirements and suggestions provided by the F1Tenth community, the physical car is based on the Traxxas Ford Fiesta 1/10th scale car. The original speed controller, receiver, and motor were removed. The speed controller and receiver were replaced with a VESC 6 MkIII software programmable speed and steering controller while the original motor was replaced with a higher performance brushless motor. In addition, the vehicle was equipped with a Nvidia Jetson TX2 for computing power and a Slamtec RPLIDAR A3 for providing LIDAR readings of the environment, which is currently the only sensor on-board the vehicle. The entire vehicle and computing components are powered by a single lithium polymer battery.

5 PRELIMINARY RESULTS

We ran 50 separate training runs of EONS with a population of 100 networks in each run and evolved for 50 epochs each. Figure 2 shows the how the fitness values evolved over time for each of the 50 different EONS runs. In this plot, we can see that all of the networks start off performing very poorly, but in some cases, the networks evolve quickly to produce better performing results. Twelve of the 50 EONS runs produce SNNs that complete two laps on all five of the training tracks. The remainder do not evolve a network in 50 epochs that completes the training process. This shows that there

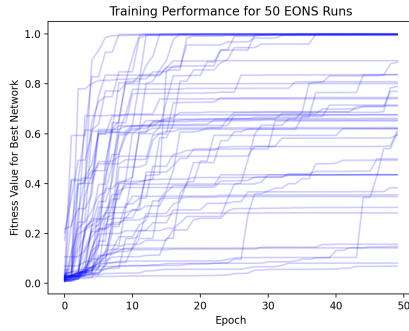


Figure 2: The fitness value of the best performing network at each epoch for all 50 EONS runs. The fitness score is the average across all five training tracks.

is significant variation in the performance of EONS for this task based on the initial starting conditions. This can be mitigated in the future by using larger population sizes.

Figure 3 shows the variation across each individual training and testing track. As can be seen in this plot, there are three training tracks that consistently perform very well (Catalunya, Budapest, and BrandsHatch), whereas the training tracks for Austin and Montreal are harder to get consistently correct. Similarly, there are testing tracks that are extremely difficult, including Yas Marina, Shanghai, and Oschersleben. The best performing network overall (shown as the red star in Figure 3) is Net 34. However, there is a well-performing network (Net 5) that performs best on Zandvoort, Sochi, Sao Paolo, and Sakhrir, but it is not actually achieve the best performance (complete two laps) on any of the training tracks.

The best network evolved with EONS, Net 34, is shown in Figure 4. This network has 39 neurons and 112 synapses. As can be seen from this figure, EONS evolves networks with complex, but sparse connectivity. There are many input to input connections (e.g., neuron 11 to neuron 3), output to output (e.g., neuron 20 to neuron 29) cycles (e.g., neurons 13 and 38 are connected), output to input (neuron 35 to 9), and so on. EONS does not restrict structure over the course of evolution, so complex structures evolve. Additionally, because the input and output neurons are also being used for computation, this network requires very few hidden neurons (three total). Because this network is very small, it can be easily and efficiently deployed to the μ Caspian hardware.

6 SUMMARY AND FUTURE WORK

Here, we demonstrate a workflow for training a neuromorphic SNN in simulation targeting a particular hardware platform. We demonstrate the success of that workflow on an autonomous racing task. Our next step is to deploy the SNN shown in Figure 4 to μ Caspian and integrate μ Caspian onto the physical car shown in Figure 1.

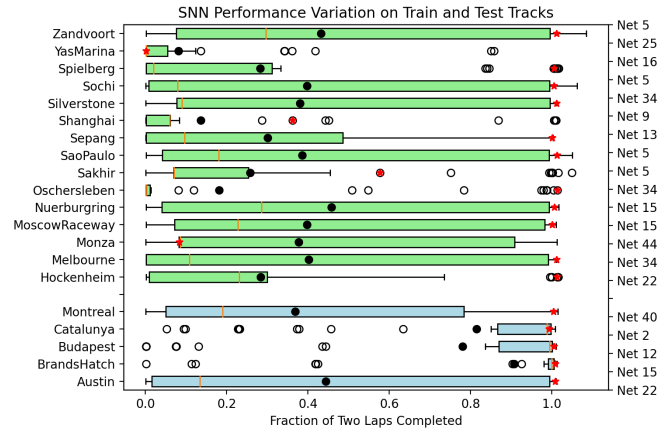


Figure 3: Variation in training and testing performance across all 50 EONS runs. Training track performance are at the bottom (shown in blue). Testing track performance are at the top (shown in green). The median value on each box plot is shown as an orange line, the mean value as a black dot, and the performance of the overall best performing network is shown as a red star. The best performing network ID for each of the individual tracks are shown along the right side of the plot.

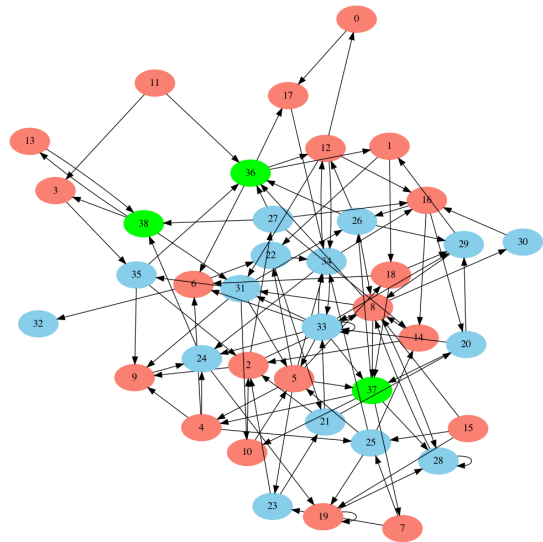


Figure 4: Best network evolved. The input neurons are shown in pink, the output neurons are shown in blue, and the hidden neurons are shown in green. Each neuron is labeled with an ID value.

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under contract number DE-AC05-00OR22725.

REFERENCES

- [1] Jonathan D Ambrose, Adam Z Foshie, Mark E Dean, James S Plank, Garrett S Rose, J Parker Mitchell, Catherine D Schuman, and Grant Bruer. 2020. GRANT: Ground-Roaming Autonomous Neuromorphic Targeter. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
- [2] Chiara Bartolozzi, Francesco Rea, Charles Clercq, Daniel B Fasnacht, Giacomo Indiveri, Michael Hofstätter, and Giorgio Metta. 2011. Embedded neuromorphic vision for humanoid robots. In *CVPR 2011 workshops*. IEEE, 129–135.
- [3] Guang Chen, Zhenshan Bing, Florian Röhrbein, Jörg Conradt, Kai Huang, Long Cheng, Zhuangyi Jiang, and Alois Knoll. 2017. Toward brain-inspired learning with the neuromorphic snake-like robot and the neurobotic platform. *IEEE Transactions on Cognitive and Developmental Systems* 11, 1 (2017), 1–12.
- [4] Guang Chen, Hu Cao, Jorg Conradt, Huajin Tang, Florian Röhrbein, and Alois Knoll. 2020. Event-based neuromorphic vision for autonomous driving: a paradigm shift for bio-inspired visual sensing and perception. *IEEE Signal Processing Magazine* 37, 4 (2020), 34–49.
- [5] Daesu Chung, Reid Hirata, T Nathan Mundhenk, Jen Ng, Rob J Peters, Eric Pichon, April Tsui, Tong Ventrice, Dirk Walther, Philip Williams, et al. 2002. A new robotics platform for neuromorphic vision: Beobots. In *International Workshop on Biologically Motivated Computer Vision*. Springer, 558–566.
- [6] Prasanna Date, Christopher D Carothers, James A Hendler, and Malik Magdon-Ismail. 2018. Efficient classification of supercomputer failures using neuromorphic computing. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 242–249.
- [7] Prasanna Date, Catherine Schuman, Bill Kay, and Thomas Potok. 2021. Neuro-morphic computing is Turing-complete. *arXiv preprint arXiv:2104.13983* (2021).
- [8] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham China, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro* 38, 1 (2018), 82–99.
- [9] Mark E Dean, Catherine D Schuman, and J Douglas Birdwell. 2014. Dynamic adaptive neural network array. In *International Conference on Unconventional Computation and Natural Computation*. Springer, 129–141.
- [10] Julien Dupeyroux. 2021. A toolbox for neuromorphic sensing in robotics. *arXiv preprint arXiv:2103.02751* (2021).
- [11] Xiao Guo, Zhenjiang Shen, Yajing Zhang, and Teng Wu. 2019. Review on the application of artificial intelligence in smart homes. *Smart Cities* 2, 3 (2019), 402–420.
- [12] Kathleen Hamilton, Prasanna Date, Bill Kay, and Catherine Schuman D. 2020. Modeling epidemic spread with spike-based models. In *International Conference on Neuromorphic Systems 2020*. 1–5.
- [13] Kathleen Hamilton, Tiffany Mintz, Prasanna Date, and Catherine D Schuman. 2020. Spike-based graph centrality measures. In *International Conference on Neuromorphic Systems 2020*. 1–8.
- [14] Tiffany Hwu, Jacob Isbell, Nicolas Oros, and Jeffrey Krichmar. 2017. A self-driving robot using deep convolutional neural networks on neuromorphic hardware. In *2017 international joint conference on neural networks (IJCNN)*. IEEE, 635–641.
- [15] Bill Kay, Prasanna Date, and Catherine Schuman. 2020. Neuromorphic graph algorithms: Extracting longest shortest paths and minimum spanning trees. In *Proceedings of the Neuro-inspired Computational Elements Workshop*. 1–6.
- [16] J Parker Mitchell, Grant Bruer, Mark E Dean, James S Plank, Garrett S Rose, and Catherine D Schuman. 2017. NeoN: Neuromorphic control for autonomous robotic navigation. In *2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*. IEEE, 136–142.
- [17] J Parker Mitchell, Mark E Dean, Grant R Bruer, James S Plank, and Garrett S Rose. 2018. DANNA 2: Dynamic adaptive neural network arrays. In *Proceedings of the International Conference on Neuromorphic Systems*. 1–6.
- [18] J Parker Mitchell, Catherine D Schuman, Robert M Patton, and Thomas E Potok. 2020. Caspian: A neuromorphic development platform. In *Proceedings of the Neuro-inspired Computational Elements Workshop*. 1–6.
- [19] J Parker Mitchell, Catherine D Schuman, and Thomas E Potok. 2020. A small, low cost event-driven architecture for spiking neural networks on fpgas. In *International Conference on Neuromorphic Systems 2020*. 1–4.
- [20] Thomas Netter and Nicolas Francheschini. 2002. A robotic aircraft that follows terrain using a neuromorphic eye. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1. IEEE, 129–134.
- [21] James S Plank, Catherine D Schuman, Grant Bruer, Mark E Dean, and Garrett S Rose. 2018. The TENNLab exploratory neuromorphic computing framework. *IEEE Letters of the Computer Society* 1, 2 (2018), 17–20.
- [22] Catherine D Schuman, J Parker Mitchell, Robert M Patton, Thomas E Potok, and James S Plank. 2020. Evolutionary optimization for neuromorphic systems. In *Proceedings of the Neuro-inspired Computational Elements Workshop*. 1–9.
- [23] Catherine D Schuman, James S Plank, Grant Bruer, and Jeremy Anantharaj. 2019. Non-traditional input encoding schemes for spiking neuromorphic systems. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–10.
- [24] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. 2017. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963* (2017).
- [25] Catherine D Schuman, Steven R Young, J Parker Mitchell, J Travis Johnston, Derek Rose, Bryan P Maldonado, and Brian C Kaul. 2020. Low Size, Weight, and Power Neuromorphic Computing to Improve Combustion Engine Efficiency. In *2020 11th International Green and Sustainable Computing Workshops (IGSC)*. IEEE, 1–8.