Building a Comprehensive Neuromorphic Platform for Remote Computation

William Severa*, Aaron J. Hill,
Craig M. Vineyard, Ryan Dellana,
Leah Reeder, Felix Wang, James B. Aimone
Sandia National Laboratories
Albuquerque, NM, 87123
Email: *wmsever@sandia.gov

Angel Yanguas-Gil
Applied Materials Division
Argonne National Laboratory
Lemont, IL 60439

***Abstract*—Remote sensing and extreme environments present a unique and critical algorithm and hardware tradeoff due to extreme size, weight and power constraints. Consequently, in many applications, systems favor centralized computation over remote computation. However, in some real-time systems (e.g. a Mars rover) latency and other communication bottlenecks force on-board processing. With traditional processor performance at a plateau, we look to brain-inspired, non-Von Neumann neuromorphic architectures to enable future capabilities, such as event detection/tracking and intelligent decision making. These cutting-edge hardware platforms generally operate at vastly improved performance-per-Watt ratios, but have suffered from niche applications, difficult interfaces, and poor integration with existing algorithms. In this paper, we discuss methods, motivated by recent results, to produce a cohesive neuromorphic system that effectively integrates novel and traditional algorithms for context-driven remote computation.**

***Keywords*—Deep Learning; Neuromorphic Computing; Remote Computation; Neuromodulation**

I. Introduction

In deployed remote systems, such as satellites, ground sensors, and aerial systems, on-board computation faces a distinct challenge of incorporating effective algorithms at low/ultra-low size, weight and power (SWaP) cost. This ever-present tradeoff has been helped historically by the exponential improvement in transistors following Moore's Law. However, processor development slowdown and the popularity of computationally intensive deep learning algorithms have motivated the need for new solutions. Neural-inspired computation looks to the brain as inspiration for how to structure algorithms as well as hardware architectures. With *morphic* meaning to have the form or structure of, neuromoprhic computer architectures strive to mimic neuron function, circuitry, and computation of the brain to offer state-of-the-art performance at milliwatt scale [1], [2], even while using conventional CMOS technology. We envision these technologies will enable high levels of computation at the sensor (Fig. 1). In practice these platforms have faced a number of critical challenges (algorithm compatibility, programming interfaces, at-scale production), but fortunately recent developments we discuss here have begun to work past these difficulties.

The remainder of the paper is structured as follows. We explore recent efforts in porting algorithms, both traditional data processing and deep learning algorithms, to neuromorphic platforms in Section II. We then discuss the use of context modulation for flexible, efficient deep learning networks in Section III. Benchmarks and forecasts for performance of various neuromorphic platforms are included in Section IV. We briefly describe the growing field of neural-inspired sensors in Section V. Finally, we provide some concluding observations and remarks in Section VI.

DISTRIBUTION STATEMENT A. Approved for public release: distribution is unlimited. Approval ID: SAND2019-0257 C

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

II. Porting Algorithms to Neuromorphic Platforms

In many application spaces, tried-and-tested classical non-learning algorithms are verified and well-understood, providing a strong level of assurance. For a neuromorphic platform to compete, it must be able to implement the functionality of these classical algorithms as well as the emerging state-of-the-art seen in deep learning.

Neural-inspired computing frameworks have largely focused on learning based algorithms and it has been difficult to perceive how neural networks could implement classical non-learning methods. However, the field is beginning to approach neurons as highly parallel, simple processors [3], and similar approaches have led to a systematic methodology to create neuromorphic-compatible classical algorithms in a variety of fields [4]–[6]. Using these methods we can build composite algorithms for neural hardware from baseline kernels such as cross-correlation [3], matrix multiplication [7], or min, max, and sorting [8]. Additionally, random walks [9] can be conducted using neural algorithms with efficient spike representations of the walkers. The simple schematic for this algorithm can either model each particle's location individually, or model each node in a graph and track the number

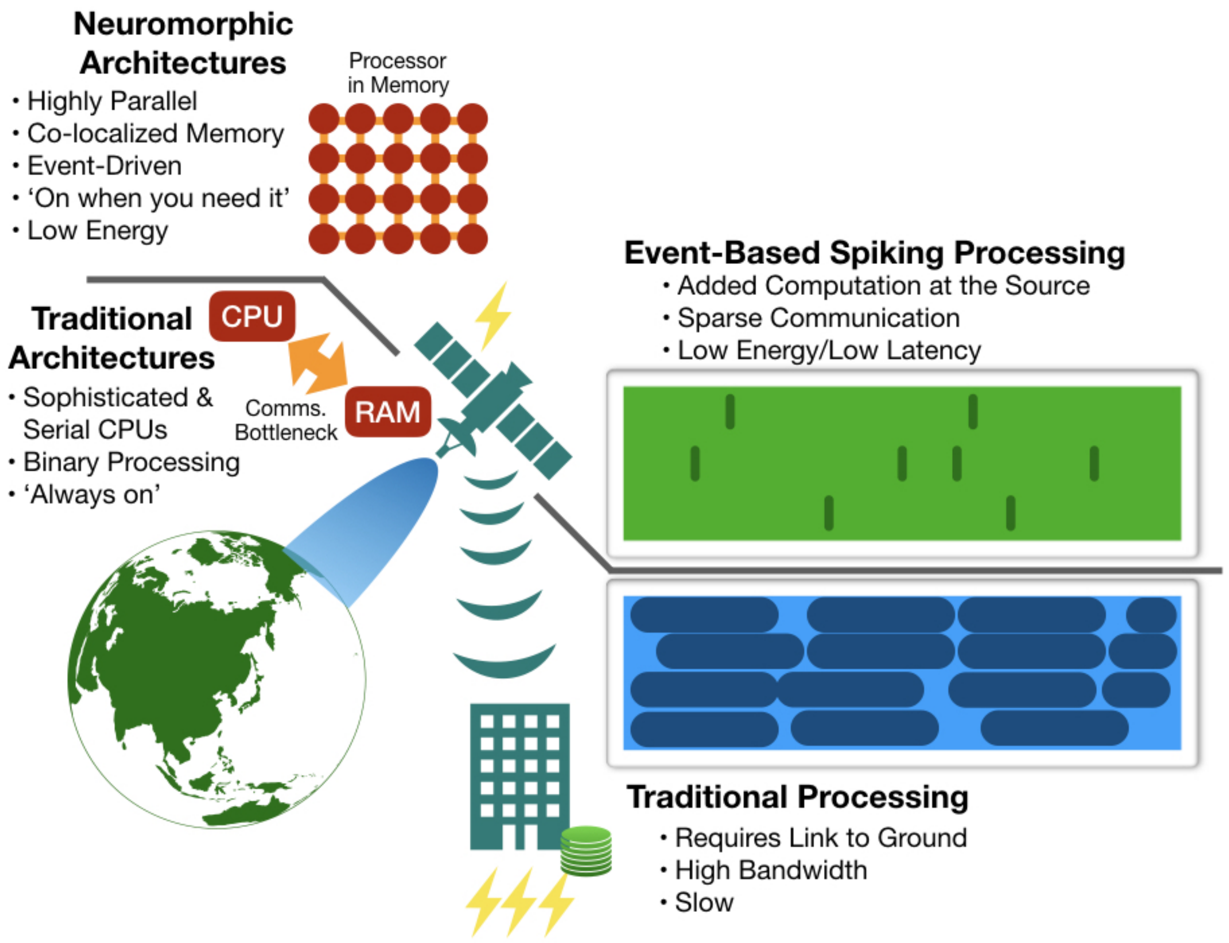

Fig. 1. Neuromorphic processors can enable ultra-low power computation at the sensor by using event-driven spiking processing. This allows advanced data, signal, and image processing in remote environments (space, underwater, aerial, etc.) while requiring lower energy and bandwidth costs. By combining processing and memory (via neurons) into a homogenous platform, neural systems can overcome the von Neumann communications bottleneck seen in traditional processors.

of particles at that location. This neural algorithm [9] can then be used in any random walk application space, such as image analysis and graph processing, with the added benefits of using neuromorphic architectures. And we remark that while for some individual kernels neuromorphic computing may not offer a substantial benefit over conventional systems, representing the entire computation as a neural algorithm may yield substantial overall application benefits by avoiding the transitions between conventional and neuromorphic hardware.

Implementing deep learning on neuromorphic poses a different challenge. The vast majority of deep learning frameworks do not operate on discrete spike-like communication that neuromorphic architectures provide. This creates difficulty when seeking to implement existing deep networks on neuromorphic hardware, requiring custom learning frameworks tuned to the specific underlying hardware [10]. However, recently developed methods allow for an easy transition from existing learning frameworks to neuromorphic hardware [11], [12]. Utilizing the Whetstone method in [11] produced classification accuracy results on MNIST [13] and CIFAR-10 [14] of 99.5% and 88.0%, respectively.

III. Context-Sensitive Deep Learning

Neuromorphic processors, such as IBM's TrueNorth and Intel's Loihi, can perform deep learning tasks at extremely low-power consumption [15]–[17]. In practice, applications are rarely single-task jobs. More commonly, applications require a combination of subtasks (e.g. event detection and navigation) to integrate towards an overall goal, and having a separate accelerator or special-purpose component for each subtask is not feasible. Instead, if a neuromorphic architecture is included in the design of a system, the neuromorphic chip must have the flexibility to perform many different tasks, preferably without being reprogrammed. This flexibility is something biological systems exhibit in nearly all situations [18], and recent work suggests that artificial neural networks can be designed to be as flexible. By incorporating neuromodulation (the idea that diffuse, network-wide inputs can adjust behavior), deep learning networks can exhibit context-dependent behavior [19]. Similar mechanisms have also been shown to adjust behaviors according to weighted goals [20]. Recently, we have implemented neuromodulation to perform context-dependent tasking. For example, we train a single network to operate under two separate contexts ('detect cars' or 'detect people'), and then in the testing phase, we can alternate between these two modes of operation without adjusting the network.

In addition to flexibility, another key motivation for incorporating context in a deep learning network is to make more efficient use of computational resources. In general, lower-level functionality (e.g. convolution kernels and simple features) may be readily shared across distinct higher-level functionality (e.g. recognizing different classes). A prime example of this is to use pre-trained networks to facilitate transfer learning (e.g. VGG, GloVe) [21]. For our goal of enabling a singular network to perform well under different contexts or modes of operation, we supplement the training process by providing a context signal to the network through a parallel pathway. This context information subsequently affects downstream processing.

We demonstrate this capability by training a network to perform superclass exclusion, where different classes to be learned may be hierarchically categorized into superclasess. As a result, lower level features that contribute to a classification are mapped differently depending on knowledge of a superclass. Here, we also find that when the superclass of the input was made available as a context signal, we were able to improve the performance of the network. In experiments training on the CIFAR-100 (images, [22]) and 20 Newsgroups (text, [14]) datasets, we achieved an increase in classification accuracy when the networks were provided with context-dependent bias. Specifically, we saw an increase in classification accuracy of 24% from a baseline of 44% to 68% on the CIFAR-100 dataset, and an increase of 18% from 50% to 68% on the 20 Newsgroups dataset.

Related to classification, a more sophisticated approach is to train a network to detect multiple classes in different modes of operation. That is, instead of the output of the network taking on one out of several classes, we have the network output take a binary decision on whether a class has been detected (e.g. in an image). Here, we use the CIFAR-10 dataset (images) and use the provided context signal to switch operation between the detection of different classes. Experimentally, we achieved high detection accuracy (target class vs. non-target classes) for each target class. Across the board, we saw detection accuracies above 90%, with the highest at 97% (for horses), all within the same network.

IV. Programming and Performance of Neuromorphic Hardware

When designing neural algorithms it is easiest to assume an unconstrained neural fabric with no limitations to neuron count or connectivity. In practice this is not true and the constraints are unique to the underlying hardware. With the vast variety of different neuromorphic hardware available, a vast variety of programming interfaces and constraints follow. For instance, IBM's TrueNorth architecture builds its programming interface on MATLAB [25], while SpiNNaker builds its interface on Python utilizing the PyNN module [26]. Further, each architecture has its own unique constraints on neuron count and connectivity, while a true neural algorithm is agnostic to these constraints. Thus, it is the algorithm implementors task to dissect the neural algorithm in a way that will fit into the underlying hardware constraints and conform to the programming language provided. For example, TrueNorth only allows a single neuron to connect to 256 other neurons which must all exist on the same core. What if the algorithm requires a neuron to connect to 1000 other neurons across multiple cores? There are certainly ways to achieve the result necessary, but it requires careful understanding of the hardware and timing dynamics of the algorithm. SpiNNaker is unique in that it does not constrain connectivity, however there are underlying memory constraints that do limit the number of connections possible in an unpredictable way.

Figure 2 depicts how several prominent neuromorphic architectures have balanced architectural tradeoffs (right) as well as captures trends in performance for various computational architectures (left). For the performance measure, we benchmarked using similar networks and data adapated to benefit each platform. Since performance can vary drastically according to, among other things, network design and input dimensionality, reported numbers are linearly normalized to a $1024 = 32 \times 32$ pixel image. Included are the STPU [27], Intel Neural Compute Stick, IBM TrueNorth [15], and SpiNNaker (48-node board) [28]. We have forecasted performance for the SpiNNaker 2 platform [29] and adapted results from literature for Intel Loihi (single chip) [30]. The networks run on SpiNNaker were trained using Whetstone [11]; the networks for TrueNorth used EEDN [10].

V. Neuromorphic Sensing

Not only can composing and computing an entire problem via a neural approach provide benefit, but the means by which input data is collected and presented to a neuromorphic processor can significantly impact the performance of a comprehensive system. Using neurons as the computational medium, inputs need to be formatted in a representation that neural networks can work with. This might entail adequate precision or how the data is represented. Spiking neuromorphic approaches communicate with a single bit, corresponding to when a neuron exceeds threshold and fires, and may include a temporal component to the representation. Transduction is a step employed by some approaches to convert an input signal into a spiking equivalent. This conversion process is non-trivial, impacting the net computation. Alternatively, directly collecting or generating data in a neural-inspired manner can alleviate the conversion step and provide other computational advantages. For example, the Dynamic Vision Sensor takes inspiration from the retina and performs change detection at the photo detector only transmitting changes in an observed scene [31], [32]. Effectively, this approach provides high-speed data transmission while natively providing spiking in-

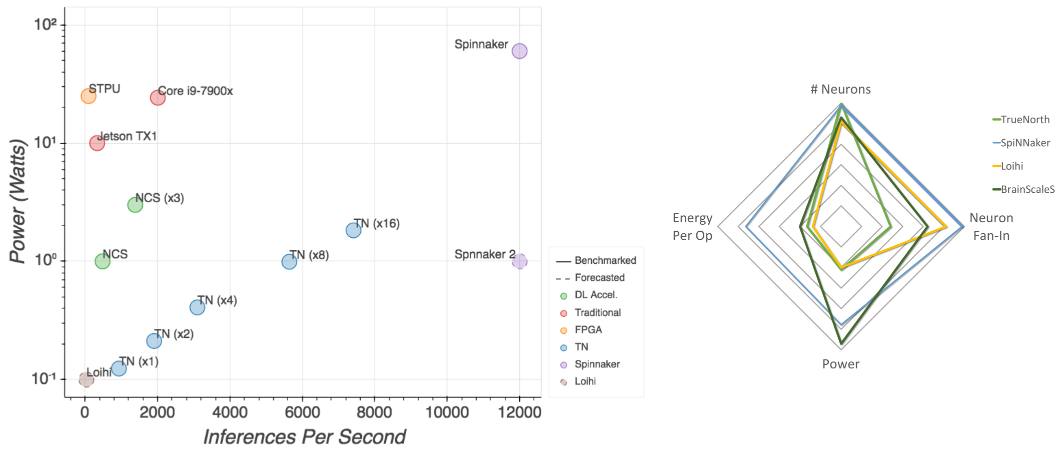

Fig. 2. The landscape of neuromorphic processors exhibits a wide variety of design tradeoffs. However, a key differentiator from traditional processing methods is an improvement in effective performance-per-Watt. **Left:** A chart showing benchmark deep learning performance of various neuromorphic and reference platforms. **Right:** The design considerations of each neural platform provides interesting tradeoffs that affect scale, energy per operation, total energy and flexibility (fan-in) [23], [24]. These design choices impact the suitability of applications appropriate for each platform.

puts a subsequent neuromorphic processor can operate upon and has been used for a number of applications including embedded gesture recognition and tracking [17], [33].

VI. Conclusion and Discussion

Consequently, while deep neural networks have risen to prominence in many computational domains, classic architectures such as CPUs and GPUs are either not ideally suited for their execution or require large computational costs making their usage prohibitive in resource constrained environments. Alternatively by developing both algorithms and architectures taking inspiration from the brain these advances together can enable comprehensive neuromorphic platforms which can build upon neuroscience inspired principles such as parallelism, sparse event-driven computation, and simplicity of computational units with complex connectivity. In effect, comprehensive neuromorphic platforms provide a path forwards for enabling sophisticated remote sensing in extreme environments while operating within size, weight, and power constraints.

References

[1] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[2] E. Neftci, "Data and power efficient intelligence with neuromorphic learning machines," *iScience*, 2018.

[3] W. Severa, O. Parekh, K. D. Carlson, C. D. James, and J. B. Aimone, "Spiking network algorithms for scientific computing," in *Rebooting Computing (ICRC), IEEE International Conference on*. IEEE, 2016, pp. 1–8.

[4] J. B. Aimone, O. Parekh, and W. Severa, "Neural computing for scientific computing applications: more than just machine learning," in *NCS*, 2017.

[5] X. Lagorce and R. Benosman, "Stick: spike time interval computational kernel, a framework for general purpose computation using neurons, precise timing, delays, and synchrony," *Neural computation*, vol. 27, no. 11, pp. 2261–2317, 2015.

[6] J. V. Monaco and M. M. Vindiola, "Integer factorization with a neuromorphic sieve," in *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–4.

[7] O. Parekh, C. A. Phillips, C. D. James, and J. B. Aimone, "Constant-depth and subcubic-size threshold circuits for matrix multiplication," in *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*. ACM, 2018, pp. 67–76.

[8] S. J. Verzi, F. Rothganger, O. D. Parekh, T.-T. Quach, N. E. Miner, C. M. Vineyard, C. D. James, and J. B. Aimone, "Computing with spikes: The advantage of fine-grained timing," *Neural computation*, pp. 1–31, 2018.

[9] W. Severa, R. Lehoucq, O. Parekh, and J. B. Aimone, "Spiking neural algorithms for markov process random walk," in *International Joint Conference on Neural Networks 2018*. IEEE, 2018.

[10] S. Esser, P. Merolla, J. Arthur, A. Cassidy, R. Appuswamy, A. Andreopoulos, D. Berg, J. McKinstry, T. Melano, D. Barch *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing. 2016," *Preprint on ArXiv. http://arxiv. org/abs/1603.08270. Accessed*, vol. 27, 2016.

[11] W. Severa, C. M. Vineyard, R. Dellana, S. J. Verzi, and J. B. Aimone, "Training deep neural networks for binary communication with the whetstone method," *Nature: Machine Intelligence*, In Press.

[12] E. Hunsberger and C. Eliasmith, "Training spiking deep networks for neuromorphic hardware," *arXiv preprint arXiv:1611.05141*, 2016.

[13] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[14] K. Lang, "Newsweeder: Learning to filter netnews," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 331–339.

[15] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[16] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[17] A. Amir, B. Taba, D. J. Berg, T. Melano, J. L. McKinstry, C. Di Nolfo, T. K. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, "A low power, fully event-based gesture recognition system." in *CVPR*, 2017, pp. 7388–7397.

[18] V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome, "Context-dependent computation by recurrent dynamics in prefrontal cortex," *nature*, vol. 503, no. 7474, p. 78, 2013.

[19] J. B. Aimone and W. M. Severa, "Context-modulation of hippocampal dynamics and deep convolutional networks," *arXiv preprint arXiv:1711.09876*, 2017.

[20] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," *arXiv preprint arXiv:1611.01779*, 2016.

[21] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[22] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[23] C. S. Thakur, J. Molin, G. Cauwenberghs, G. Indiveri, K. Kumar, N. Qiao, J. Schemmel, R. Wang, E. Chicca, J. O. Hasler *et al.*, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *arXiv preprint arXiv:1805.08932*, 2018.

[24] S. Furber, "Large-scale neuromorphic computing systems," *Journal of neural engineering*, vol. 13, no. 5, p. 051001, 2016.

[25] A. Amir, P. Datta, W. P. Risk, A. S. Cassidy, J. A. Kusnitz, S. K. Esser, A. Andreopoulos, T. M. Wong, M. Flickner, R. Alvarez-Icaza *et al.*, "Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–10.

[26] A. P. Davison, D. Brüderle, J. M. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "Pynn: a common interface for neuronal network simulators," *Frontiers in neuroinformatics*, vol. 2, p. 11, 2009.

[27] A. J. Hill, J. W. Donaldson, F. H. Rothganger, C. M. Vineyard, D. R. Follet, P. L. Follett, M. R. Smith, S. J. Verzi, W. Severa, F. Wang, J. B. Aimone, J. H. Naegle, and C. D. James, "A spike-timing neuromorphic processor," in *Proceedings of the IEEE Conference on Rebooting Computing*, 2017.

[28] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, 2013.

[29] C. Liu, G. Bellec, B. Vogginger, D. Kappel, J. Partzsch, F. Neumärker, S. Höppner, W. Maass, S. B. Furber, R. Legenstein *et al.*, "Memory-efficient deep learning on a spinnaker 2 prototype," *Frontiers in neuroscience*, vol. 12, 2018.

[30] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," *arXiv preprint arXiv:1812.01739*, 2018.

[31] T. Delbrück, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 2426–2429.

[32] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128$\times$ 128 1.5% contrast sensitivity 0.9% fpn 3 μs latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers." *J. Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, 2013.

[33] D. Drazen, P. Lichtsteiner, P. Häfliger, T. Delbrück, and A. Jensen, "Toward real-time particle tracking using an event-based dynamic vision sensor," *Experiments in Fluids*, vol. 51, no. 5, p. 1465, 2011.