

Event-driven Robust Fitting on Neuromorphic Hardware

Tam Ngoc-Bang Nguyen¹, Anh-Dzung Doan¹, Zhipeng Cai², Tat-Jun Chin¹
¹ Australian Institute for Machine Learning, The University of Adelaide,
² Intel Labs

{tam.nb.nguyen, dzung.doan, tat-jun.chin}@adelaide.edu.au

czptc2h@gmail.com

Abstract

Robust fitting of geometric models is a fundamental task in many computer vision pipelines. Numerous innovations have been produced on the topic, from improving the efficiency and accuracy of random sampling heuristics to generating novel theoretical insights that underpin new approaches with mathematical guarantees. However, one aspect of robust fitting that has received little attention is energy efficiency. This performance metric has become critical as high energy consumption is a growing concern for AI adoption. In this paper, we explore energy-efficient robust fitting via the neuromorphic computing paradigm. Specifically, we designed a novel spiking neural network for robust fitting on real neuromorphic hardware, the Intel Loihi 2. Enabling this are novel event-driven formulations of model estimation that allow robust fitting to be implemented in the unique architecture of Loihi 2, and algorithmic strategies to alleviate the current limited precision and instruction set of the hardware. Results show that our neuromorphic robust fitting consumes only a fraction (15%) of the energy required to run the established robust fitting algorithm on a standard CPU to equivalent accuracy.

1. Introduction

Many computer vision pipelines, ranging from visual SLAM, 3D reconstruction to image stitching, need to estimate geometric models from noisy and outlier-contaminated measurements that occur when operating in real-world environments [48]. Often, this is achieved by optimising the model parameters according to a robust criterion defined over the data, known as *robust fitting*.

Random sampling heuristics such as RANSAC [29] and its variants [46] are established techniques for robust fitting. Such methods usually deliver satisfactory outcomes, but provide little insights on the veracity of the results [64]. On the other hand, techniques that rely on mathematical programming can provide some quality guarantees, but are

typically too slow to be practical on real data [17].

Significant research has been devoted into robust fitting for computer vision. This includes improving the speed, accuracy and generalisability of random sampling methods [8, 21, 57] and deriving theoretical insights to better inform the design and usage of mathematical programming techniques (*e.g.*, adapting to special cases, relaxing the guarantees) [42, 55]. Machine learning approaches that can leverage statistics in training data to hypothesis sampling have also been developed [11]. Recently, quantum computing has also been explored for robust fitting [18].

An aspect of robust fitting algorithm research that has received comparatively little attention is *energy efficiency*. With the rapidly rising energy consumption of AI systems becoming a concern [22], it is vital to develop vision algorithms that are energy-efficient. We argue that devising low-power alternatives for core components such as robust fitting is an essential step to enable ambitious low-power end-to-end 3D vision pipelines.

In this paper, we explore *energy-efficient robust fitting* via the neuromorphic computing, which is a bio-inspired computational model where a network of processing units called spiking neurons asynchronously send spike-based messages to each other [60]. Such a structure is called a spiking neural network (SNN). Due to the massive parallelism, stochastic behaviour, and event-driven computing, SNNs promise higher energy efficiency than conventional computing, including artificial neural networks (ANN) [35].

Currently available neuromorphic processors that can implement SNNs include IBM TrueNorth [49], SpiN-Naker [34] and Intel Loihi 1 and 2 [24, 53, 62]. Comprehensive experiments [25] indicate the much higher energy efficiency of the neuromorphic devices, which underlines their potential in reducing the energy consumption of data centres [3] and their application on embodied AI systems [44].

Contributions We develop an SNN that can conduct robust fitting on neuromorphic hardware, specifically Intel Loihi 2 [62]. Underpinning our SNN are novel event-driven formulations of core steps in robust fitting, namely minimal

subset sampling, model estimation and model verification, that make the problem amenable to a neuromorphic treatment. We also propose strategies to mitigate the current limitations on precision and instruction set of Loihi 2.

When simulated on the CPU, results of our SNN on synthetic data and real datasets verify its correctness and competitive accuracy relative to state-of-the-art robust fitting methods. Importantly, experiments on Loihi 2 illustrate the vastly superior energy efficiency of our SNN, in that it consumes only a fraction (15%) of the energy required by established methods on the CPU.

2. Related work

2.1. Computing paradigms for robust fitting

Due to the crucial role of robust fitting in computer vision, various computing paradigms and hardware platforms have been explored to accelerate their execution.

FPGAs offer highly parallel computing capabilities, making them well-suited for real-time robust fitting tasks. Several works have explored robust fitting on FPGAs, leveraging the mechanism of parallel hypothesis evaluation to improve performance [67–69]. However, challenges such as limited on-chip memory, as well as difficulties in programming, optimising, and debugging algorithms, make FPGA implementations nontrivial [61].

By devising differentiable versions of robust fitting and reformulating the problem as a machine learning task [12, 13, 70], GPUs have been adopted to conduct robust fitting. However, inferencing large neural networks on GPUs can be energy intensive [5, 31, 32]. Also, machine learning approaches suffer from generalizability issues if the testing data distribution differs from the training data distribution.

Recently, robust fitting using quantum computing has gained significant interest within the research community [17, 26, 72]. These studies have shown promising potential for accelerating robust fitting processes. However, the quantum approaches remain in an experimental phase, with testing limited to small-scale data due to current hardware limitations. Furthermore, quantum computers have high energy demands due to cooling requirements [45, 51].

Unlike the above, we show that neuromorphic computing offers both high energy efficiency and usability.

2.2. Neuromorphic computing for optimisation

Research on algorithms in neuromorphic computing can be broadly categorized into two main directions: machine learning and optimization [25, 60]. Learning approaches focus on converting pre-trained ANNs into corresponding SNNs for on-chip inference, directly learning SNN parameters through training equivalent proxy ANNs, and approximation of backpropagation on neuromorphic hardware. In the optimization domain, the spike-based temporal process-

ing characteristics of SNNs are exploited to develop solutions for optimization tasks. Our work belongs to the latter.

The temporal dynamics of SNNs have been actively explored in combinatorial and continuous optimization. In integer domains, a majority of handcrafted SNNs were designed for constraint satisfaction problems (CSPs), including travelling salesman problem, Sudoku, Boolean satisfiability and graph coloring [10, 30, 40, 54, 71]. The core idea behind these approaches is to encode CSP variables and constraints into an SNN topology. The SNN-based CSP solvers iteratively explore and refine the solution space, until seeking an assignment for variables that satisfy all constraints of the original CSP problem. Another line of focus in SNN-based combinatorial solvers is quadratic unconstrained binary optimization (QUBO) [6, 23, 28, 50, 56]. While the primary objective of CSP solvers is to find feasible assignments to a combinatorial optimization problem, QUBO aims to find optimal solutions, where the inherent spike-based temporal dynamics of SNNs can be viewed as iterative solvers for optimization problems [44, 56, 63].

2.3. Neuromorphic computing for vision

Neuromorphic computing is receiving increased attention in the vision community [15, 59, 66]. [15, 59] designed SNN-based for optical flow estimation from event stream data. [66] integrated Loihi chip as a co-processor for computing angular error for drone control tasks. Neuromorphic computing has been explored in visual place recognition [37] and SLAM [41]. It is worthwhile to note that not all the works have tested on actual neuromorphic processors. Moreover, few have paid attention to robust fitting.

3. Preliminaries

We first review the problem formulation for robust fitting before giving a brief overview of SNNs and Intel Loihi 2.

3.1. Robust fitting

We describe the procedure for robustly fitting a linear regression model, which is the specific problem targeted by our proposed SNN. This does not reduce the applicability of our ideas since many geometric models can be linearized [36, Chap. 4]. Moreover, our main aim is to establish the viability of neuromorphic robust fitting, and its extension to nonlinear models is left as future work.

Given a set of N measurements $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, we wish to estimate the linear equation $y = \mathbf{x}^T \boldsymbol{\theta}$. The least squares (LS) solution is

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^d} \sum_{i=1}^N (y_i - \mathbf{x}_i^T \boldsymbol{\theta})^2 = \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^d} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|_2^2 \quad (1)$$

where $\mathbf{X} \in \mathbb{R}^{N \times d}$ and $\mathbf{y} \in \mathbb{R}^N$ are obtained by vertically stacking $\{\mathbf{x}_i^T\}_{i=1}^N$ and $\{y_i\}_{i=1}^N$, respectively. If \mathcal{D} contains

outliers, $\hat{\theta}$ will be biased. Instead, robust fitting aims to find the model θ that minimizes the objective function

$$\sum_{i=1}^N \rho(|y_i - \mathbf{x}_i^T \theta|), \quad (2)$$

where ρ is a robust loss [74]. Widely used in vision is

$$\rho(r) = \begin{cases} 1 & r > \epsilon_{\text{inlier}}, \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

where ϵ_{inlier} is the inlier threshold, hence, minimizing (2) with (3) is equivalent to maximizing the inlier count

$$\Psi(\theta) = \sum_{i=1}^N \mathbb{I}(|y_i - \mathbf{x}_i^T \theta| \leq \epsilon_{\text{inlier}}) \quad (4)$$

of θ , where $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the input condition is true and 0 otherwise. The value $\Psi(\theta)$ is called the consensus of θ [16], and the resultant estimate is robust to outliers provided ϵ_{inlier} was selected appropriately.

Generating model hypotheses θ by sampling minimal subsets is a successful approach for robust fitting [47]. Basically, three main steps are repetitively executed:

1. Sample a d -subset $\mathcal{S} \subset \mathcal{D}$ (minimal subset) of the data.
2. Estimate a model hypothesis by LS fitting (1) on \mathcal{S} .
3. Evaluate the quality of the hypothesis using (2).

At termination, the algorithm returns the model with the best objective value. If the number of repetitions K is large enough, at least one all-inlier minimal subset \mathcal{S} will be sampled, leading to a robust estimate of the model.

3.2. SNN

A neuromorphic algorithm can be designed as an SNN that is then executed on a neuromorphic computer [60]. Conceptually, an SNN consists of a set of N spiking neurons, where each pair of neurons is connected via synapses. The synaptic connection strengths are represented by a weight matrix $\mathbf{W} = [w_{ij}] \in \mathbb{R}^{N \times N}$, where $w_{ij} = 0$ indicates the absence of a connection between the i -th and j -th neurons.

Each spiking neuron comprises internal states that accumulate input stimuli over time, and emits spikes only when predefined conditions are met. Upon spiking, it transmits a spike signal to connected neurons and may enter into a refractory (inactive) period. Well-known spiking neuron models include Leaky Integrate-and-Fire [33], Resonate-and-Fire [38] and Izhikevich [39], though the SNN framework is flexible enough to allow custom neurons with specific computations. The synaptic weights and neuronal processing define the problem that is solved by the SNN.

3.3. Intel Loihi 2

Each Loihi 2 chip [53] houses 128 asynchronous neuromorphic cores (neuro cores) which can simulate up to 8192

stateful and parallel spiking neurons. At each neuro core, the ingress spikes from other cores enter a *Synapse* block, where a dense/ sparse matrix-vector multiplication or convolution is performed; see Fig. 1. The result is then accumulated and passed as inputs to the *Neuron* block. Here, stateful neuron programs are executed and 24-bit spike messages are generated and routed to other neuro cores [24, 44]. The SNN architecture can be designed using *Dense*, *Sparse*, and *Convolution* synaptic configurations of 8-bit weights, and the neuron models can be customised through assembly code with up to 24-bit internal states.

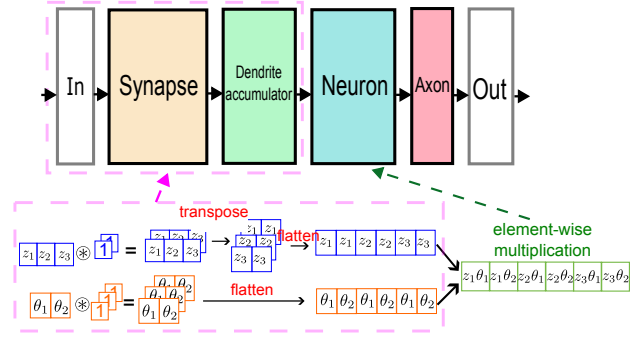


Figure 1. The top diagram shows a high-level schematic of a neuro core in Loihi 2. Note that Neuron block is programmable. The bottom diagram shows our technique to emulate matrix-matrix multiplication and how it is mapped onto the neuro core. See Fig. AA Supplementary for more details.

4. Neuromorphic robust fitting

Here, we describe the proposed SNN for neuromorphic robust fitting, including the mathematical underpinnings, spiking neuron designs, and hardware implementation details. Note that while spiking neurons are conceptually asynchronous or event-driven [60], the operations of neurons below are described using discrete timesteps t to give more intuitive depiction of time evolution. This matches the programming at the *neuron level* in real neuromorphic computers such as Loihi 2, which are fully digital devices [24]. However, it should be reminded that t is the *algorithmic time* rather than a global synchronous clock.

4.1. SNN for robust fitting

Fig. 2 illustrates the proposed SNN for robust fitting, called *NeuroRF*, which consists of the following neurons:

- N RandomSampling neurons $\{z_i\}_{i=1}^N$.
- d ModelHypothesis neurons $\{\theta_j\}_{j=1}^d$.
- Nd Auxiliary neurons $\{\theta'_{i,j}\}_{i=1, \dots, N}^{j=1, \dots, d}$.
- N ComputeResidual neurons $\{c_j\}_{j=1}^N$.
- 1 InlierCounter neuron Σ .

Algorithm 1 defines the internal arithmetic operations that evolve the state of each neuron. Note that, unlike in

the case of classical (von Neumann) computers, there is no clear main body and parent-subfunction relationships, though the neurons influence each other by sending output spikes through the interconnections.

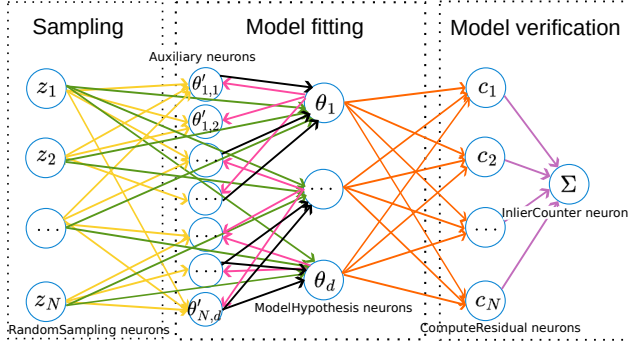


Figure 2. The proposed NeuroRF SNN.

The behaviour of NeuroRF will be described in more detail below via its main operations, while details of implementation on Loihi 2 will be provided in Sec. 4.2.

4.1.1. Random sampling

As the name suggests, the role of the RandomSampling neurons is conducting random sampling of the input data. Each z_i is a binary variable, where $z_i = 1$ means that the i -th point (\mathbf{x}_i, y_i) is selected and $z_i = 0$ means otherwise.

For a d -dimensional model, each neuron emits a spike with probability d/N . This is encoded in a simple dynamic by comparing a randomly generated number with a constant d/N . However, since the neurons sample independently, they may not collectively select exactly d points. Since we conduct estimation using gradient descent (details to follow), over- or under-sampling the points (including selecting no points) do not cause numerical issues.

4.1.2. Model hypothesis generation

The LS objective (1) can be written in the quadratic form

$$f(\theta) = \theta^T \mathbf{Q} \theta + \mathbf{p}^T \theta, \quad (5)$$

where $\mathbf{Q} = \mathbf{X}^T \mathbf{X}$ and $\mathbf{p}^T = -\mathbf{y}^T \mathbf{X}$. Since $f(\theta)$ is convex, it justifies using gradient descent (GD) to solve least squares, where the first-order gradient is

$$\nabla f(\theta) = \mathbf{Q} \theta + \mathbf{p}, \quad (6)$$

and we iteratively update θ via

$$\theta^{(t)} = \theta^{(t-1)} - \alpha \left(\mathbf{Q} \theta^{(t-1)} + \mathbf{p} \right) \quad (7)$$

with α being the step size or learning rate for M iterations.

Previous studies have shown that the spike-based temporal dynamics of SNNs align with the behaviour of classical

iterative solvers [44, 52, 56, 63]. In fact, the GD iteration (7) readily lends itself as the dynamic equations for the ModelHypothesis neurons, where each evolves according to

$$\theta_j^{(t)} = \theta_j^{(t-1)} - \alpha \left(\mathbf{q}_j \theta^{(t-1)} + p_j \right), \quad (8)$$

with \mathbf{q}_j being the j -th row of \mathbf{Q} and p_j being the j -th element of \mathbf{p} . Conceptually, \mathbf{q}_j and p_j are respectively the synaptic weights and bias leading into the j -th neuron.

However, a direct application of (8) is problematic for our aims, since the data for estimation is encoded in \mathbf{Q} and \mathbf{p} , which need to be repetitively sampled at each robust fitting iteration, while the synaptic weights and biases are constant in the Synapse block of Loihi 2 during operation.

We propose an algebraic manipulation that enables fully neuromorphic random sampling and model hypothesis generation. Collecting the RandomSampling states in a vector $\mathbf{z} = [z_1, z_2, \dots, z_N]^T$, we “lift” the gradient (6) as

$$\nabla f(\theta, \mathbf{z}) = \mathbf{Q}' \theta' + \mathbf{P}' \mathbf{z} \quad (9)$$

where

$$\begin{aligned} \mathbf{Q}' &= [\mathbf{x}_1 \mathbf{x}_1^T \quad \mathbf{x}_2 \mathbf{x}_2^T \quad \dots \quad \mathbf{x}_N \mathbf{x}_N^T] \in \mathbb{R}^{d \times Nd}, \\ \theta' &= \text{vec}(\mathbf{z} \theta^T) = \mathbf{z} \otimes \theta \in \mathbb{R}^{Nd}, \\ \mathbf{P}' &= [-y_1 \mathbf{x}_1 \quad -y_2 \mathbf{x}_2 \quad \dots \quad -y_N \mathbf{x}_N] \in \mathbb{R}^{d \times N}. \end{aligned} \quad (10)$$

Note that the \otimes is the Kronecker product. The lifting allows the gradient to be a function of the data selection via \mathbf{z} . The GD update is now also dependent on \mathbf{z} , *i.e.*,

$$\theta^{(t)} = \theta^{(t-1)} - \alpha \left(\mathbf{Q}' \theta'^{(t-1)} + \mathbf{P}' \mathbf{z} \right). \quad (11)$$

See Sec. A Supplementary for details of the derivation. Note that the extreme cases ($\mathbf{z} = \mathbf{1}$ and $\mathbf{z} = \mathbf{0}$) do not cause numerical issues, since the gradient reduces to $\nabla f(\theta)$ and $\mathbf{0}$ respectively (no operation in the latter case).

The expansion (10) creates Nd Auxiliary neurons θ' , which are unrolled into $\{\theta'_{i,j}\}_{i=1, \dots, N}^{j=1, \dots, d}$. Each Auxiliary neuron evolves according to

$$\theta'_{i,j}{}^{(t)} = z_i^{(t-1)} \theta_j^{(t-1)}, \quad (12)$$

which behaves as a coupling of a pair of RandomSampling and ModelHypothesis neurons. More importantly, the ModelHypothesis neurons now follow the dynamical equation

$$\theta_j^{(t)} = \theta_j^{(t-1)} - \alpha \left(\mathbf{q}'_j \theta'^{(t-1)} + \mathbf{p}'_j \mathbf{z} \right), \quad (13)$$

where \mathbf{q}'_j is the j -th row of \mathbf{Q}' and \mathbf{p}'_j is the j -th row of \mathbf{P}' . The synaptic weights and biases leading into the j -th ModelHypothesis Neuron θ_j are now *constant*.

Algorithm 1 NeuroRF algorithm

RandomSampling layer

Require: Switching probability $prob = \frac{d}{N}$
Require: $\tau = 2M + 4$, $\mathcal{L} = K * \tau$
1: Initialize $k \leftarrow 0$, $counter \leftarrow 0$
2: Initialize $\mathbf{z}^{(0)} \leftarrow \mathbf{0}_N$
3: **for** $t = 1, 2, \dots, \mathcal{L}$ **do**
4: $counter += 1$
5: $next_sampling_idx = \tau * k + 1$
6: **if** $counter = next_sampling_idx$ **then**
7: $k += 1$
8: $\gamma^{(t)} = \text{rand}[0, 1)$
9: $\mathbf{z}^{(t)} = \text{prob} \geq \gamma^{(t)}$
10: **else**
11: $\mathbf{z}^{(t)} = \mathbf{z}^{(t-1)}$
12: Send $\mathbf{z}^{(t)}$ to connected Auxiliary and ModelHypothesis layers.

Auxiliary layer

Require: Connection matrices $\mathbf{F}_d, \mathbf{F}_N$
Require: $\tau = 2M + 4$, $\mathcal{L} = K * \tau$
1: Initialize $\boldsymbol{\theta}'^{(0)} \leftarrow \mathbf{0}_{Nd}$
2: Initialize $k \leftarrow 1$, $counter \leftarrow 0$
3: **for** $t = 1, 2, \dots, \mathcal{L}$ **do**
4: $\mathbf{z}_{in}^{(t)} = \mathbf{F}_d \otimes \mathbf{z}^{(t-1)}$
5: $\boldsymbol{\theta}_{in}^{(t)} = \mathbf{F}_N \otimes \boldsymbol{\theta}^{(t-1)}$
6: $counter += 1$
7: $next_sampling_idx = \tau * k + 1$
8: **if** $counter = next_sampling_idx$ **then**
9: $k += 1$
10: $\boldsymbol{\theta}'^{(t)} = \mathbf{0}_{Nd}$
11: **else**
12: $\boldsymbol{\theta}'^{(t)} = \mathbf{z}_{in}^{(t)} * \boldsymbol{\theta}_{in}^{(t)}$
13: Send $\boldsymbol{\theta}'^{(t)}$ to connected ModelHypothesis layer.

4.1.3. Model verification

The ComputeResidual and InlierCounter neurons calculate the residuals and consensus (4) of the current model estimate (states of the ModelHypothesis neurons). These are straightforward, and the reader is referred to Algorithm 1.

4.2. Mitigating hardware limitations

To implement NeuroRF on Loihi 2 [62], the SNN architecture and synaptic weights are defined on a host CPU, then mapped to the neuro cores of Loihi 2 via the Lava framework [1]. Consensus sets found are read out and returned to the host CPU once the algorithm finishes. This probe is performed by special embedded processors that can access the states of neuro cores [15, 27].

However, Loihi 2 has several constraints: 8-bit weights

Algorithm 1 NeuroRF algorithm (cont.)

ModelHypothesis layer

Require: Connection matrices \mathbf{Q}' , \mathbf{P}' and learning rate α
Require: $\tau = 2M + 4$, $\mathcal{L} = K * \tau$
1: Initialize $\boldsymbol{\theta}^{(0)} \leftarrow \mathbf{0}_d$
2: Initialize $k \leftarrow 1$, $counter \leftarrow 0$
3: **for** $t = 1, 2, \dots, \mathcal{L}$ **do**
4: $\mathbf{a}_{in}^{(t)} = \mathbf{Q}'\boldsymbol{\theta}'^{(t-1)}$
5: $\text{bias}^{(t)} = \mathbf{P}'\mathbf{z}^{(t-1)}$
6: $counter += 1$
7: $next_sampling_idx = \tau * k + 1$
8: **if** $counter = next_sampling_idx$ **then**
9: $k += 1$
10: $\boldsymbol{\theta}^{(t)} = \mathbf{0}_d$
11: **else**
12: **if** $counter$ is odd **then**
13: $\text{total_grad}^{(t)} = \mathbf{a}_{in}^{(t)} + \text{bias}^{(t)}$
14: $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \alpha * \text{total_grad}^{(t)}$
15: **else**
16: $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)}$
17: Send $\boldsymbol{\theta}^{(t)}$ to Auxiliary layer

ComputeResidual layer

Require: Connection matrix \mathbf{X} , bias vector \mathbf{y} and inlier threshold ϵ_{inlier}
Require: $\tau = 2M + 4$, $\mathcal{L} = K * \tau$
1: Initialize $\mathbf{c} \leftarrow \mathbf{0}_N$
2: **for** $t = 1, 2, \dots, \mathcal{L}$ **do**
3: $\mathbf{a}_{in}^{(t)} = \mathbf{X}\boldsymbol{\theta}^{(t-1)}$
4: $\text{res}^{(t)} = \mathbf{a}_{in}^{(t)} - \mathbf{y}$
5: $\mathbf{c}^{(t)} = |\text{res}^{(t)}| \leq \epsilon_{inlier}$
6: Send $\mathbf{c}^{(t)}$ to InlierCounter neuron

InlierCounter neuron

Require: Connection matrix $\mathbf{1}_N^T$
Require: $\tau = 2M + 4$, $\mathcal{L} = K * \tau$
1: **for** $t = 1, 2, \dots, \mathcal{L}$ **do**
2: $\Psi^{(t)} = \mathbf{1}_N^T \mathbf{c}^{(t-1)}$

which can limit the instance size for testing, no support for floating-point arithmetic, a reduced instruction set (no division) and synapse configurations with limited customisability [2, 44, 56]. Here, we present strategies to mitigate these constraints. Note that the limitations may be solved in future iterations of Loihi, and do not detract from the mathematical and logical consistencies of NeuroRF.

Integer arithmetic All data \mathcal{D} is assumed integer or converted to integer from rational numbers. The optimal model estimate $\boldsymbol{\theta}$ may not be integral, but NeuroRF is aimed at

finding the best integer solution. As we will show in Sec. 5, the integer approximation does not significantly reduce the quality (consensus size) obtained, as observed in other applications on Loihi 2 [44, 63].

Random sampling Loihi 2’s pseudo-random generator (PRG) is restricted to either 16 or 24 bit integers [56]. We describe our approach to using 16-bit PRG, but the idea is similar for the 24-bit PRG. Let $LFSR \in [0, 2^{16} - 1]$ be the Loihi 2-generated random number. The required threshold $\gamma \in [0, 1)$ is mathematically obtained as

$$\gamma = \frac{LFSR}{2^{16} - 1} \approx \frac{LFSR}{2^{16}}. \quad (14)$$

From Algorithm 1, the switching condition for a Random-Sampling neuron is mathematically defined as

$$d > \frac{N * LFSR}{2^{16}}. \quad (15)$$

We approximate the division using the arithmetic right shift, leading to the switching logic

$$d > ((N * LFSR) \ggg 16). \quad (16)$$

GD update step size The GD process (11) requires a step size parameter α , which should be sufficiently small to not cause divergence. To allow a floating-point α , we use fixed-point representation [44, 63] for storing α . Specifically, an $\alpha \in \mathbb{R}$ can be converted to a fixed-point counterpart

$$\bar{\alpha} = \text{ceil}(\alpha * 2^\beta), \quad (17)$$

where $\beta \in \mathbb{Z}$ is a tuned constant. Here $\bar{\alpha}$ is rounded up to the nearest integer. The update (11) can be approximated as

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \left(\mathbf{Q}'\boldsymbol{\theta}^{(t-1)} + \mathbf{P}'\mathbf{z} \right) \frac{\bar{\alpha}}{2^\beta}. \quad (18)$$

Using the arithmetic right shift, the GD update becomes

$$\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)} - \left[\left(\mathbf{Q}'\boldsymbol{\theta}^{(t-1)} + \mathbf{P}'\mathbf{z} \right) * \bar{\alpha} \right] \ggg \beta \quad (19)$$

This approximation at each update step could cause overall drift of the solution [44, 63], thus the number of steps should be kept small to avoid large build-up of errors.

Matrix-matrix multiplications On Loihi 2, neurons must connect through Synapse (See Fig 1), where an interaction $\boldsymbol{\theta}' = \text{vec}(\mathbf{z}\boldsymbol{\theta}^T)$ is not feasible. To realise the Auxiliary neural dynamics, we introduce two synaptic connection matrices into $\boldsymbol{\theta}'$

$$\boldsymbol{\theta}' = \text{vec} \left[\left(\mathbf{1}_d \mathbf{z}^T \right)^T \right] * \text{vec} \left(\mathbf{1}_N \boldsymbol{\theta}^T \right). \quad (20)$$

Here $\mathbf{1}_d$ and $\mathbf{1}_N$ can be viewed as synaptic weights of matrix-matrix multiplication Synapses, which are not supported on Loihi 2. To overcome this limitation, we emulate matrix-matrix multiplication with *Conv Synapse*

$$\left(\mathbf{1}_d \mathbf{z}^T \right) = \mathbf{F}_d \otimes \mathbf{z}, \quad \left(\mathbf{1}_N \boldsymbol{\theta}^T \right) = \mathbf{F}_N \otimes \boldsymbol{\theta}, \quad (21)$$

where \otimes is a convolution operation. $\mathbf{F}_N \in \mathbb{R}^{N \times 1 \times 1}$ and $\mathbf{F}_d \in \mathbb{R}^{d \times 1 \times 1}$ are respectively the set of N and d (1×1) convolutional filters. See Fig. 1 for a visual explanation of our approach to emulate matrix-matrix multiplication and how this process is mapped onto neuro cores.

5. Results

Our experiments focused on establishing the correctness of NeuroRF, its performance on a neuromorphic chip, and its potential in lowering the energy cost of robust fitting.

5.1. Method variants

Two different variants of NeuroRF were used:

- **NeuroRF-CPU**: NeuroRF was implemented in Lava [1] with 64-bit floating-point arithmetic and x86-64 instruction set, and run on an Intel Core i7-11700K. This simulated NeuroRF on a CPU.
- **NeuroRF-Loihi**: NeuroRF was implemented on Loihi 2 (specifically on the single-chip Oheo Gulch board [2]) using Lava-Loihi 0.7.0 [1] and Loihi assembly language [53]. Note that Loihi 2 had limited precision and instruction set, which we mitigated in Sec. 4.2.

5.2. Correctness of NeuroRF

We first validate NeuroRF on robust linear regression via synthetic data. For each data instance, we randomly generated a true model $\boldsymbol{\theta}^*$ and N independent measurements $\{\mathbf{x}_i\}_{i=1}^N$. Each y_i was then computed based on linear model $\mathbf{y} = \boldsymbol{\theta}^T \mathbf{x}$ and perturbed with Gaussian noise of $\sigma_{\text{inlier}} = 0.1$. To simulate outliers, $\eta\%$ data points were randomly corrupted with Gaussian noise of $\sigma_{\text{outlier}} = 1.5$. The inlier threshold ϵ_{inlier} was set to 0.5.

To rigorously test our method, we generated 65 problem instances of varying levels of difficulty, with 5 instances produced for each combination of (N, d, η) :

- $N \in [100, 200, 400, 300, 500]$ with $d = 8$ and outlier ratio $\eta = 20$.
- $d \in [2, 3, 6, 8]$ with $N = 200$ and $\eta = 20$.
- $\eta \in [10, 20, 30, 40, 50, 60]$ with $N = 200$ and $d = 8$.

We compared NeuroRF and a classical robust fitting algorithm RANSAC [29], denoted as **RS-CPU**, running on an Intel Core i7-11700K. The number of iterations K was set to 300 for both methods, while $M = 200$ and $\alpha = 0.02$ were configured for NeuroRF-CPU. Since the goal here is to verify the correctness of our SNN on synthetic data, it is sufficient to compare against RS-CPU. We will benchmark against more advanced methods on real data in Sec. 5.4.

To evaluate robust fitting accuracy, we employed the normalized Euclidean distance between the ground-truth and output models, θ_{gt} and θ_{est} , from a robust fitting method:

$$100 * \|\theta_{gt} - \theta_{est}\|_2 / \|\theta_{gt}\|_2. \quad (22)$$

We recorded the average and std. dev. of the normalized distance over 10 trials for each method.

Fig. 3 plots the normalized distance across the N , d and η variations. NeuroRF-CPU achieved comparable performance to RS-CPU on the three setups, which confirms the algorithmic soundness of NeuroRF and its effectiveness in handling high-dimensional problems. Also, see Sec. C Supp. for runtime results. Note that the runtime figures here were from *simulating* NeuroRF on a CPU, and hence are not reflective of runtime on Loihi 2.

5.3. Performance on neuromorphic hardware

Due to the relatively low capacity of Loihi 2, we tested NeuroRF-Loihi on line fitting problems only ($d = 2$).

Since the synaptic weights are 8 bits only, we generated synthetic data as follows: in each instance, the true model θ^* was sampled from $\{-10, -9, \dots, 9, 10\}^2$, from which $N \in \{10, 20\}$ integer points that satisfy the linear relation were generated. All points were randomly corrupted with noise of ± 1 , before $\eta \in \{10, 20, \dots, 50\}$ percent were selected and corrupted further with noise of ± 4 . For each (N, η) combination, we generated 5 instances, leading to a total of 50 line fitting problems.

We compared NeuroRF-Loihi and RS-CPU; the more advanced methods (Sec. 5.4) were not more energy-efficient than RS-CPU. For both methods, ϵ_{inlier} was set to 4 and the number of iterations K was set to 100. For NeuroRF-Loihi, we also set $M = 200$, $\alpha = 0.02$ and $\beta = 10$.

We recorded the distance (22), runtime and dynamic energy consumption of the methods. Results were averaged over 10 trials and the std. dev. was also provided in the case of consensus size. For RS-CPU, runtime was obtained with *time* python module while the energy consumption was measured with Intel’s RAPL technology via pyJoules [4]. For NeuroRF-Loihi, the energy consumption and runtime were obtained via the built-in Loihi 2 Profiler.

Fig. 4 shows results for $N = 20$ (plots for $N = 10$ are available in Sec. D Supp). Fig. 4a shows the normalized distance in percentage of NeuroRF-Loihi and RS-CPU, which clearly indicates that both are on par in terms of solution quality. Fig. 4b shows that NeuroRF-Loihi consumed about only 15% of the energy required by RS-CPU, a definitive proof of the much higher energy-efficiency of NeuroRF-Loihi. However, Fig. 4c shows that the runtime of NeuroRF-Loihi was greater than RS-CPU. This was probably because NeuroRF-Loihi employed GD for LS, while RS-CPU solved LS analytically. Also, the speed of each Loihi 2 node is not as high as a cutting-edge CPU.

5.4. Affine image registration

We illustrate the applicability of NeuroRF to robust fitting on real visual data through an image registration problem.

An affine transformation \mathbf{H}_A warps a point \mathbf{x} in one image to a corresponding point \mathbf{x}' in another image via

$$\mathbf{x}' = \mathbf{H}_A \tilde{\mathbf{x}}, \quad (23)$$

where $\mathbf{x}' = (x', y')$ and $\tilde{\mathbf{x}} = [\mathbf{x}^T \ 1] = (x, y, 1)$ is \mathbf{x} in homogeneous coordinates. The 2×3 affinity matrix \mathbf{H}_A can be estimated by solving a linear system constructed from 3 point correspondences [36, Chap. 2]. Each correspondence $\langle \mathbf{x}'_i, \mathbf{x}_i \rangle$ contributes two equations, forming a system of six equations to estimate 6 parameters. The model dimension d of the affine image registration problem is hence 6. When outliers are present in the correspondences, robust estimation methods can be applied with a minimal subset size of 3. For the conversion to (4) and how to apply our method to affine transformation, see Sec. B Supp.

We created affine registration instances from 8 scenes of VGG Dataset¹. In total, 40 image pairs were selected and SIFT feature matches were extracted with VLFeat toolbox [65], which were pruned using Lowe’s 2nd nearest neighbor test. See Fig. 5 for samples and qualitative results.

Metric As most VGG ground-truth homographies are near-affine, the estimated affine matrix \mathbf{H}_A can be lifted to a full homography $\mathbf{H}_{est} \in \mathbb{R}^{3 \times 3}$ by appending a projective row [36, Chap. 2]. We follow the evaluation protocol in [43, 58, 73] and compute the area under the cumulative error curve (AUC). For each pair, we project the four corners using ground-truth and estimated homographies, compute the corner error, and report AUC up to a 10-pixel threshold. See, e.g., [58, Sec. 5.2], for details of this metric.

Competitors We compared the performance of NeuroRF-CPU against RANSAC [29] and the advanced methods: LO-RANSAC [20], PROSAC [19], Graph-cut RANSAC [7] and MAGSAC++ [9]. Except RANSAC, which was implemented with numpy, other methods were based on OpenCV [14]. Also, LS refinement was executed on the final consensus for all random sampling methods. We set the hyperparameters $K = 300$ for all 6 solvers, while $M = 200$ and $\alpha = 0.02$ were selected for NeuroRF-CPU.

Results We report the AUC at 5 and 10 pixels for these methods on two subsets: the near-affine subset (30 image pairs) and the entire dataset. As shown in Tab. 1, our NeuroRF-CPU achieved competitive AUCs compared to RS-CPU and OpenCV’s RANSAC on both subsets. Note that the full dataset includes two non-affine scenes, which likely contribute to higher corner projection errors and lower AUC due to their projective transformations. Qualitative results from our NeuroRF-CPU are provided in Fig. 5.

¹<https://www.robots.ox.ac.uk/vgg/research/affine/>

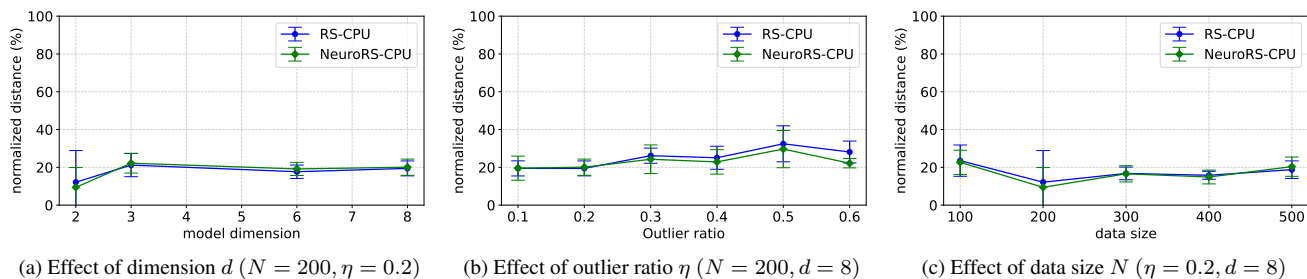


Figure 3. Normalized Euclidean distance (%) across various levels of difficulty. Results were averaged over 10 trials for each method.

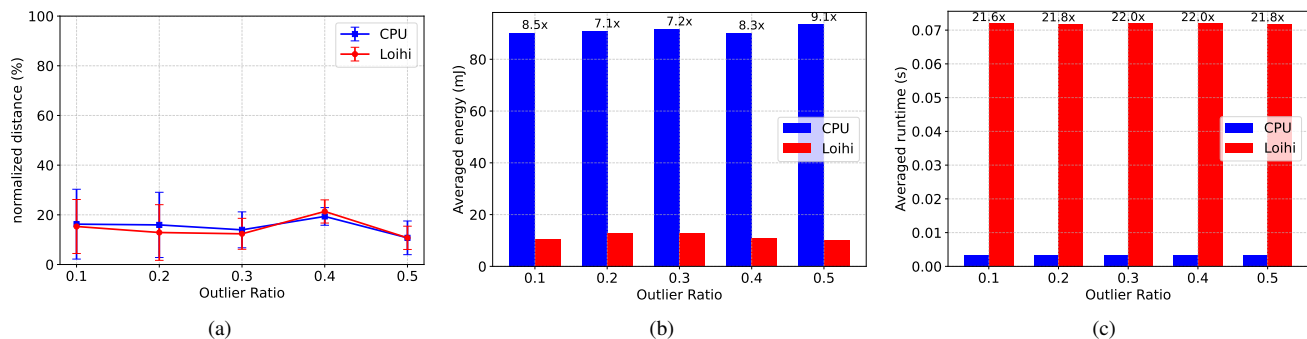
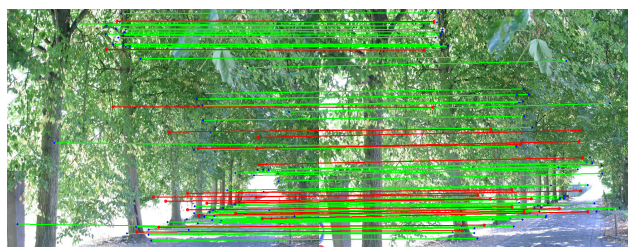
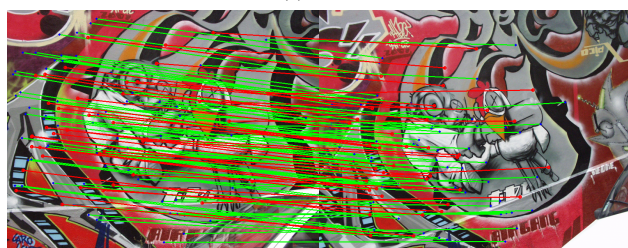


Figure 4. Performance on neuromorphic hardware. (a) Normalized Euclidean distance (%), (b) average dynamic energy consumption and (c) average runtime of NeuroRS-Loihi and RS-CPU on synthetic line fitting instances with $N = 20$ points, plotted against outlier rate.



(a) Trees



(b) Graf

Figure 5. Green and red lines represent inliers and outliers found by NeuroRS-CPU on two affine image registration instances.

6. Conclusions and future work

We designed an SNN for robust fitting and implemented it on a real neuromorphic processor. Despite the limitations of the current neuromorphic hardware, by carefully mitigating the constraints, we were able to establish the viability of neuromorphic robust fitting and its superior energy

Methods	Near-affine only		Full dataset	
	AUC@5	AUC@10	AUC@5	AUC@10
RANSAC [29]	0.396	0.61	0.297	0.458
MAGSAC++ [9]	0.393	0.599	0.294	0.449
GC-RANSAC [7]	0.397	0.61	0.298	0.458
PROSAC [19]	0.396	0.608	0.297	0.456
LO-RANSAC [20]	0.397	0.611	0.298	0.458
NeuroRF-CPU	0.396	0.608	0.297	0.456

Table 1. AUC evaluated at 5 and 10 pixels. Values closer to 1 indicate better performance.

efficiency compared to the original CPU version.

6.1. Future work

Our current SNN is catered to fitting linear models only. Extending to nonlinear models commonly encountered in computer vision will be useful.

The low capacity and precision of Loihi 2 prevent NeuroRF from being competitive against SOTA robust fitting methods. An interesting future work will be to implement NeuroRF on a neuromorphic cluster [3] and benchmark against SOTA methods.

Last but not least, building more advanced applications based on NeuroRF, such as augmented reality, visual odometry, SLAM and 3D mapping will be of interest.

Acknowledgement

We acknowledge Intel Labs and the Intel Neuromorphic Research Community (INRC) for granting access to Loihi 2 and providing technical support. Tat-Jun Chin is SmartSat CRC Professorial Chair of Sentient Satellites.

References

- [1] Lava Neuromorphic Computing framework. <https://github.com/lava-nc/lava>. 5, 6
- [2] Highlights of Loihi 2 instruction sets. <https://download.intel.com/newsroom/2021/new-technologies/neuromorphic-computing-loihi-2-brief.pdf>, . 5, 6
- [3] Intel Builds World’s Largest Neuromorphic System to Enable More Sustainable AI. <https://www.intel.com/content/www/us/en/newsroom/news/intel-builds-worlds-largest-neuromorphic-system.html>, . 1, 8
- [4] pyJoules’s documentation. <https://pyjoules.readthedocs.io/en/latest/>. 7
- [5] Negar Alizadeh and Fernando Castor. Green AI: A preliminary empirical study on energy consumption in dl models across different runtime infrastructures. In *Proceedings of the IEEE/ACM International Conference on AI Engineering-Software Engineering for AI*, 2024. 2
- [6] Md Zahangir Alom, Brian Van Essen, Adam T Moody, David Peter Widemann, and Tarek M Taha. Quadratic unconstrained binary optimization (qubo) on neuromorphic computing system. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3922–3929. IEEE, 2017. 2
- [7] Daniel Barath and Jiří Matas. Graph-cut ransac. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6733–6741, 2018. 7, 8
- [8] Daniel Barath and Jiří Matas. Graph-cut ransac. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1
- [9] Daniel Barath, Jana Noskova, Maksym Ivashchkin, and Jiri Matas. Magsac++, a fast, reliable and accurate robust estimator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1304–1312, 2020. 7, 8
- [10] Jonathan Binas, Giacomo Indiveri, and Michael Pfeiffer. Spiking analog vlsi neuron assemblies as constraint satisfaction problem solvers. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2094–2097. IEEE, 2016. 2
- [11] Eric Brachmann and Carsten Rother. Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses . In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4321–4330, 2019. 1
- [12] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 2
- [13] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2
- [14] Gary Bradski, Adrian Kaehler, et al. Opencv. *Dr. Dobb’s journal of software tools*, 3(2), 2000. 7
- [15] Stefano Chiavazza, Svea Marie Meyer, and Yulia Sandamirskaya. Low-latency monocular depth estimation using event timing on neuromorphic hardware. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4071–4080, 2023. 2, 5
- [16] Tat-Jun Chin and David Suter. *The maximum consensus problem: recent algorithmic advances*. Morgan & Claypool Publishers, 2017. 3
- [17] Tat-Jun Chin, Zhipeng Cai, and Frank Neumann. Robust fitting in computer vision: Easy or hard? In *Proceedings of the European Conference on Computer Vision*, 2018. 1, 2
- [18] Tat-Jun Chin, David Suter, Shin-Fang Ch’ng, and James Quach. Quantum robust fitting. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 1
- [19] Ondrej Chum and Jiri Matas. Matching with prosac-progressive sample consensus. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, pages 220–226. IEEE, 2005. 7, 8
- [20] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint pattern recognition symposium*, pages 236–243. Springer, 2003. 7, 8
- [21] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, 2003. 1
- [22] Ariel Cohen. AI is pushing the world toward an energy crisis. <https://www.forbes.com/sites/arielcohen/2024/05/23/ai-is-pushing-the-world-towards-an-energy-crisis/>. 1
- [23] Kevin Corder, John V Monaco, and Manuel M Vindiola. Solving vertex cover via ising model on a neuromorphic processor. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018. 2
- [24] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham China, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018. 1, 3
- [25] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R Risbud. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021. 1, 2
- [26] Anh-Dzung Doan, Michele Sasdelli, David Suter, and Tat-Jun Chin. A hybrid quantum-classical algorithm for robust fitting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2
- [27] Matthew Evanusa, Yulia Sandamirskaya, et al. Event-based attention and tracking on neuromorphic hardware. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 5
- [28] Yan Fang and Ashwin Sanjay Lele. Solving quadratic unconstrained binary optimization with collaborative spiking

- neural networks. In *2022 IEEE International Conference on Rebooting Computing (ICRC)*, pages 84–88. IEEE, 2022. 2
- [29] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 1, 6, 7, 8
- [30] Gabriel A Fonseca Guerra and Steve B Furber. Using stochastic spiking neural networks on spinnaker to solve constraint satisfaction problems. *Frontiers in neuroscience*, 11: 714, 2017. 2
- [31] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, and Håkan Grahn. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 2019. 2
- [32] Stefanos Georgiou, Maria Kechagia, Tushar Sharma, Federica Sarro, and Ying Zou. Green AI: Do deep learning frameworks have different costs? In *Proceedings of the International Conference on Software Engineering*, 2022. 2
- [33] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002. 3
- [34] Hector A Gonzalez, Jiaxin Huang, Florian Kelber, Khaleelulla Khan Nazeer, Tim Langer, Chen Liu, Matthias Lohrmann, Amirhossein Rostami, Mark Schöne, Bernhard Vogginger, et al. Spinnaker2: A large-scale neuromorphic system for event-based and asynchronous machine learning. *arXiv preprint arXiv:2401.04491*, 2024. 1
- [35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>. 1
- [36] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2, 7
- [37] Somayeh Hussaini, Michael Milford, and Tobias Fischer. Applications of spiking neural networks in visual place recognition. *IEEE Transactions on Robotics*, 2025. 2
- [38] Eugene M Izhikevich. Resonate-and-fire neurons. *Neural networks*, 14(6-7):883–894, 2001. 3
- [39] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003. 3
- [40] Zeno Jonke, Stefan Habenschuss, and Wolfgang Maass. Solving constraint satisfaction problems with networks of spiking neurons. *Frontiers in neuroscience*, 10:118, 2016. 2
- [41] Raphaela Kreiser, Alpha Renner, Yulia Sandamirskaya, and Panin Pienroj. Pose estimation and map formation with spiking neural networks: towards neuromorphic slam. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2159–2166, 2018. 2
- [42] Huu Le, Tat-Jun Chin, Anders Eriksson, Thanh-Toan Do, and David Suter. Deterministic approximate methods for maximum consensus robust fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 1
- [43] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17627–17638, 2023. 7
- [44] Ashish Rao Mangalore, Gabriel Andres Fonseca, Sumedh R Risbud, Philipp Stratmann, and Andreas Wild. Neuromorphic quadratic programming for efficient and scalable model predictive control: Towards advancing speed and energy efficiency in robotic control. *IEEE Robotics & Automation Magazine*, 2024. 1, 2, 3, 4, 5, 6
- [45] Michael James Martin, Caroline Hughes, Gilberto Moreno, Eric B Jones, David Sickinger, Sreekanth Narumanchi, and Ray Grout. Energy use in quantum data centers: Scaling the impact of computer architecture, qubit performance, size, and thermal parameters. *IEEE Transactions on Sustainable Computing*, 2022. 2
- [46] José María Martínez-Otseta, Itsaso Rodríguez-Moreno, Iñigo Mendiadua, and Basilio Sierra. Ransac for robotic applications: A survey. *Sensors*, 23(1), 2023. 1
- [47] Matthew S. Mayo and J. Brian Gray. Elemental subsets: The building blocks of regression. *The American Statistician*, 51(2):122–129, 1997. 3
- [48] Peter Meer. Robust estimation techniques. *Computer Vision: A Reference Guide*, 2020. 1
- [49] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014. 1
- [50] Susan M Mniszewski. Graph partitioning as quadratic unconstrained binary optimization (qubo) on spiking neuromorphic hardware. In *Proceedings of the International Conference on Neuromorphic Systems*, pages 1–5, 2019. 2
- [51] Engineering National Academies of Sciences, Medicine, et al. *Quantum computing: progress and prospects*. 2018. 2
- [52] Tam Nguyen, Anh-Dzung Doan, Tat-Jun Chin, et al. Slack-free spiking neural network formulation for hypergraph minimum vertex cover. *Advances in Neural Information Processing Systems*, 37:66410–66430, 2025. 4
- [53] Garrick Orchard, E Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T Sommer, and Mike Davies. Efficient neuromorphic signal processing with loihi 2. In *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 254–259. IEEE, 2021. 1, 3, 6
- [54] Christoph Ostrau, Christian Klarhorst, Michael Thies, and Ulrich Rückert. Comparing neuromorphic systems by solving sudoku problems. In *2019 International Conference on High Performance Computing & Simulation (HPCS)*, pages 521–527. IEEE, 2019. 2
- [55] Trung Thanh Pham, Tat-Jun Chin, Konrad Schindler, and David Suter. Interacting geometric priors for robust multimodel fitting. *IEEE Transactions on Image Processing*, 2014. 1
- [56] Alessandro Pierro, Philipp Stratmann, Gabriel Andres Fonseca Guerra, Sumedh Risbud, Timothy Shea, Ashish Rao Mangalore, and Andreas Wild. Solving qubo on the loihi 2 neuromorphic processor. *arXiv preprint arXiv:2408.03076*, 2024. 2, 4, 5, 6
- [57] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. USAC: A universal framework for

- random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012. 1
- [58] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 7
- [59] Yannick Schnider, Stanisław Woźniak, Mathias Gehrig, Jules Lecomte, Axel Von Arnim, Luca Benini, Davide Scaramuzza, and Angeliki Pantazi. Neuromorphic optical flow and real-time implementation with event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4129–4138, 2023. 2
- [60] Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Prasanna Date, and Bill Kay. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022. 1, 2, 3
- [61] Kah Phooi Seng, Paik Jen Lee, and Li Minn Ang. Embedded intelligence on fpga: Survey, applications and challenges. *Electronics*, 2021. 2
- [62] Sumit Bam Shrestha, Jonathan Timcheck, Paxon Frady, Leobardo Campos-Macias, and Mike Davies. Efficient video and audio processing with loihi 2. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 13481–13485. IEEE, 2024. 1, 5
- [63] Bradley H Theilman and James B Aimone. Solving sparse finite element problems on neuromorphic hardware. *arXiv preprint arXiv:2501.10526*, 2025. 2, 4, 6
- [64] Quoc Huy Tran, Tat-Jun Chin, Wojciech Chojnacki, and David Suter. Sampling minimal subsets with large spans for robust estimation. *International Journal of Computer Vision*, 2014. 1
- [65] Andrea Vedaldi and Brian Fulkerson. VLFeat: An open and portable library of computer vision algorithms, 2012. 7
- [66] 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)*, pages 103–109. IEEE, 2021. 2
- [67] John Vourvoulakis, John Lygouras, and John Kalomiros. Acceleration of ransac algorithm for images with affine transformation. In *IEEE International Conference on Imaging Systems and Techniques*, 2016. 2
- [68] John Vourvoulakis, John Kalomiros, and John Lygouras. Fpga accelerator for real-time sift matching with ransac support. *Microprocessors and Microsystems*, 2017.
- [69] John Vourvoulakis, John Kalomiros, and John Lygouras. Fpga-based architecture of a real-time sift matcher and ransac algorithm for robotic vision applications. *Multimedia Tools and Applications*, 2018. 2
- [70] Tong Wei, Yash Patel, Alexander Shekhovtsov, Jiri Matas, and Daniel Barath. Generalized differentiable ransac. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 2
- [71] Chris Yakopcic, Nayim Rahman, Tanvir Atahary, Tarek M Taha, and Scott Douglass. Leveraging the manycore architecture of the loihi spiking processor to perform quasi-complete constraint satisfaction. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020. 2
- [72] Frances Fengyi Yang, Michele Sasdelli, and Tat-Jun Chin. Robust fitting on a gate quantum computer. In *European Conference on Computer Vision*, pages 120–138. Springer, 2024. 2
- [73] Yuyang Zhang, Jinge Wang, Shibiao Xu, Xiao Liu, and Xiaopeng Zhang. Mlifeat: Multi-level information fusion based deep local features. In *Proceedings of the Asian conference on computer vision*, 2020. 7
- [74] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing*, 15(1):59–76, 1997. 3