Computational Engineering and Technology Innovations

www.ceti.reapress.com

Comput. Eng. Technol. Innov. Vol. 3, No. 4 (2026) 265–285.

Paper Type: Original Article

A Reproducible Neuromorphic Tactile Architecture with Local Plasticity and Force Feedback

Mei Lin Nakamura[1,*], Pavel D. Sokolov[2], Adaeze O. Okonkwo[3]

[1] Khoury College of Computer Sciences, Northeastern University, Boston, MA, USA; m.nakamura@northeastern.edu.
[2] Department of Electrical and Computer Engineering, Cyprus University of Technology, Limassol, Cyprus; m.nakamura@northeastern.edu.
[3] Department of Computer Science, University of Cape Town, Cape Town, South Africa; m.nakamura@northeastern.edu.

Citation:

Received: 06 February 2026 *Revised: 16 April 2026* *Accepted: 18 July 2026*	Nakamura, M. L., Sokolov, P. D., & Okonkwo, A. O. (2026). A reproducible neuromorphic tactile architecture with local plasticity and force feedback. *Computational Engineering and Technology Innovations, 3*(4), 265-285.

Abstract

Event-based touch can reduce communication and neural processing, but its usefulness depends on the reconstruction error, learning activity and overhead of the complete processing path. We present a reproducible computational architecture that connects these quantities through explicit interfaces. A delta encoder supplies ON and OFF tactile events to a Leaky Integrate-and-Fire (LIF) array with bounded, trace-based Spike-Timing-Dependent Plasticity (STDP). A separate aggregate-force channel supplies a conventional feedback controller. We derive the encoder error bound, establish exact equivalence between clocked and lazy state evolution under specified neuron assumptions, and obtain a force-tracking bound for a first-order contact model. A cost model includes encoding, synaptic accumulation, local updates, state maintenance, wake-up overhead and queue service. Across 150 constructed tactile workloads, clocked and lazy execution produced identical spikes, with a maximum state discrepancy of 1.33×10^{-15}. A grid of 4,050 force-control runs quantified the accuracy–communication trade-off. At a 0.05 N force increment and 0.005 N bounded noise, event feedback used 94.9 packets per six-second trial on average and achieved a steady tracking Root-Mean-Square Error (RMSE) of 0.08227 N, compared with 0.08304 N for periodic feedback. Energy calculations exposed a regime in which sparse clocked execution cost less than event gating because wake-up overhead exceeded avoided state updates. These results establish a modular reference design and quantitative operating conditions. The evidence consists of mathematical analysis and software simulation; the local plasticity module was evaluated independently of the force controller. Silicon measurements and learned manipulation policies remain subsequent validation stages.

Keywords: Neuromorphic tactile sensing, Local synaptic plasticity, Event encoding, Force feedback, Very-large-scale integration architecture, Reproducible simulation.

1 | Introduction

Robotic contact produces information at several temporal scales. Initial contact and slip can generate short bursts, while a maintained load can remain nearly constant. A tactile processing system must preserve the

Corresponding Author: m.nakamura@northeastern.edu
https://doi.org/10.48314/ceti.v3i4.96
Licensee System Analytics. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

changes needed for action without spending the same communication and computation on every quiet interval. Neuromorphic engineering offered a physical and computational basis for such processing by placing dynamical state and computation close to the sensory stream [1]. In embodied systems, the relevant unit of analysis extends from the sensing surface through perception and control to interaction with the environment [2]. The practical question is therefore how much information an event representation discards, how that loss affects feedback, and which implementation costs sparsity actually avoids.

Tactile sensing makes this question unusually concrete. An event represents a change at a location and time; a force controller often requires an estimate of an absolute physical quantity. A learning array may benefit from temporally correlated spikes, whereas a controller must remain interpretable when that learning is incomplete. Reviews of neural electronic skin [3] and neuromorphic touch [4] described a broad range of sensors, encoders and processing approaches. Their diversity makes an explicit interface contract essential. An architecture drawing alone does not specify the reconstruction convention, the treatment of simultaneous spikes, the timing of a weight update, or the service capacity needed during a contact burst.

The same difficulty appears in energy comparisons. Reducing the number of nonzero synaptic operations can leave acquisition, trace maintenance, memory access and static power largely unchanged. Local learning adds operations as well as adaptability. Algorithm–hardware co-design consequently has to account for the workload and its execution policy together [5], [6]. NeuroBench formalized the distinction between hardware-independent algorithm costs and deployed system measurements [7]. For robotic systems, the 2026 framework of D'Angelo et al. further connected benchmarking to physical tasks, robot platforms and interaction metrics [8]. These developments motivate reporting a computational design at its actual evidence level while making the transition to hardware and robot experiments technically specific.

We investigate an architecture with two coordinated functions. A tactile event stream drives an adaptive Spiking Neural Network (SNN) built from Leaky Integrate-and-Fire (LIF) neurons, whose synapses change through local Spike-Timing-Dependent Plasticity (STDP). An independently calibrated aggregate-force stream drives a sampled feedback controller through delta-coded reconstruction. This arrangement gives learning and feedback different responsibilities. STDP updates the sensory representation; the force controller supplies the analyzed contact response. In the experiments reported here, the representation does not generate motor commands. This division permits exact tests of the neural implementation and a separate test of the effect of event reconstruction on control.

The contribution is the joint specification and evaluation of these interfaces. We establish an error bound for the encoder and a conditional tracking bound for the contact model, specify a causal local update rule with a clear simultaneous-event convention, and prove when idle neural ticks can be skipped without changing the mathematical trajectory. We then connect measured workload counts to transparent energy and queue models. Newly executed simulations reveal both useful operating regions and failure cases: A coarse encoder can silence learning, a burst can overflow a queue despite a modest average rate, and event gating can lose to a clocked sparse implementation when wake-up cost is appreciable. The accompanying code, configuration grid and generated outputs make each result reproducible.

2|Literature Review

2.1|Sensory Representation and Embodied Evaluation

The literature progressed from circuit principles toward integrated sensory and embodied computation, although these strands continue in parallel. Bartolozzi et al. [2] situated neuromorphic computation within the coupled requirements of perception and motor control. Mead described adaptive analog computation through the physical properties of silicon [1]. Indiveri et al. [9] subsequently compared silicon neuron circuits and showed how application requirements shape the choice of dynamics and implementation. Thakur et al. [10] surveyed large-scale spiking processors across distinct architectural approaches. The transition relevant to tactile robotics concerns the movement from a feasible neural primitive to an observable task in a closed

interaction loop. *Table 1* summarizes this trajectory as an organizing interpretation, rather than a claim that one phase replaced the preceding one.

Table 1. Development of the principal design questions and evidence types.

Period	Design Question	Representative Evidence
1990–2011	Physical neural primitives	Adaptive analog systems and silicon neuron circuits [1], [9]
1998–2015	Local temporal learning	Timing-dependent synaptic changes, competition and unsupervised recognition [11–13]
2014–2020	Scalable processing and robot-facing sensing	Integrated spiking processors and visual–tactile perception [14–17]
2020–2026	Reusable tactile data and representations	Tactile digits, edge orientation and expanded alphanumeric recognition [18–20]
2025–2026	Comparable algorithm and embodied evidence	Algorithm and system metrics, followed by physical interaction benchmarks [7], [8]

Electronic skin research developed distributed transduction and processing mechanisms that can encode spatially extensive contact signals [3]. Patil et al. [21] distinguished direct sensor–neuromorphic integration from architectures that separate sensing and processing. This distinction affects what can be removed by sparsity: An asynchronous sensing surface can avoid some acquisition work, whereas a periodically sampled sensor followed by a delta encoder still pays for every acquisition tick. The present implementation uses the latter arrangement. Its operation model therefore retains the sample and comparison cost even when no tactile event is transmitted.

Temporal encoding also changes the learning problem. Gallego et al. [22] explained how event sensors expose asynchronous changes rather than frames and how event processing requires representations adapted to that output. In touch, Liu et al. [4] reviewed the connections among biological mechanisms, sensor technologies and spike encoding. Aitsam et al. [23] examined neuromorphic computation in interactive robotics, including the integration problem across sensing, neural processing and behavior. These sources support studying temporal interfaces, while the particular force and queue bounds developed below follow from the specified reference model.

2.2 | Local Plasticity and Tactile Learning

Bi and Poo [11] demonstrated the dependence of synaptic modification on the relative timing of pre- and postsynaptic activity in cultured hippocampal neurons. Song et al. [12] analyzed how timing-dependent plasticity can induce competition among inputs. Morrison et al. [24] classified phenomenological plasticity models and clarified the consequences of pairing conventions, traces and weight dependence. For engineering, the important consequence is that the phrase STDP does not uniquely define an executable update. All-to-all versus nearest-neighbor pairing, simultaneous spikes and clipping order can produce different trajectories.

Unsupervised spiking recognition also relies on the surrounding network architecture. Diehl and Cook [13] combined STDP with inhibition and adaptive thresholds for handwritten-digit recognition. Their results concern the complete recognition system and cannot be assigned to a pair-update rule in isolation. Surrogate-gradient methods provide a different route to task-trained spiking networks, using an explicit optimization objective [25]. These approaches answer different questions: local STDP specifies a synaptic adaptation mechanism, whereas supervised training connects representations to a stated target. Our array deliberately isolates the local mechanism and tests its execution semantics. It does not infer task learning from weight change alone.

Macdonald et al. [19] provided a close tactile precedent by combining an event-based optical fingertip, unsupervised STDP processing and a Three-Nearest-Neighbor (3-NN) readout for edge orientation.

Taunyazov et al. [17] combined visual and tactile events in robot-facing container classification and rotational slip detection. These studies made the sensory task and readout explicit. Visuo-haptic integration more broadly requires appropriate data collection, complementary representations and a task-specific interpretation of the fused signals [26]. The present reference design evaluates tactile processing without an added attention or visual branch; extending it to multimodal control would require corresponding sensor streams and trained readout objectives.

Dataset scope remains decisive. Spiking Tactile Modified National Institute of Standards and Technology (ST-MNIST) comprises event-based tactile recordings of handwritten digits and supplies a benchmark for tactile recognition [18]. The 2026 Spiking Tactile Extended Modified National Institute of Standards and Technology (STEMNIST) preprint expanded the recognition setting to alphanumeric classes and released new tactile recordings and baseline experiments [20]. Neither recognition label is a grasp-success observation. A controller study requires actions, plant states or observable interaction outcomes in addition to sensory samples. Accordingly, these datasets inform the proposed evaluation path but are not presented as data used in the new force-control experiments.

2.3 | Hardware Locality and the Cost of Adaptation

At the level of Very-Large-Scale Integration (VLSI), TrueNorth demonstrated a large spiking integrated circuit with a scalable communication network [14]. Loihi supported programmable synaptic learning rules within a manycore spiking architecture [15]. ODIN demonstrated compact digital online learning using Spike-Driven Synaptic Plasticity (SDSP), which is a different learning rule from the pair-based STDP used here [16]. These examples establish feasible implementation directions. They also demonstrate why a published number for one processor cannot serve as the energy coefficient of an unrelated design without matching the primitive, precision, operating point and accounting boundary.

Algorithm and hardware overviews emphasized co-design across these layers [5], [6]. Device selection introduces another layer of constraints. Choi et al. [27] discussed memristive synapses and neurons through their device structures, switching principles and neural functions. Sehgal et al. [28] compared memory technologies for artificial synapses and described peripheral overhead, precision and device nonidealities. A dense storage medium therefore does not determine the energy of a full learning path. The present model uses technology-neutral counters and separately declared illustrative coefficients. An eventual digital or mixed-signal implementation would replace those coefficients with synthesis or measurement under the same workload.

Dynamic sparsity is similarly a joint property of input statistics, the algorithm and the implementation. Zhou et al. [29] organized forms of dynamic sparsity and identified the need for co-design to exploit them. Chowdhury et al. [30] discussed coordinated sensing, algorithms and hardware for robotic vision. The architectural lesson transfers to touch: Avoiding zero-input arithmetic is useful only if the scheduler, state representation and peripheral circuits can capitalize on the omitted work. *Table 2* compares representative studies by their actual evaluation objects. The scope column records what each experiment or framework addresses, rather than treating a different research objective as a defect.

Table 2. Comparison of representative studies by task and evidence boundary.

Study and Mechanism	Variation Considered	Data or Case	Evaluation	Scope Relevant Here
See et al. [18] Tactile event dataset	Human writing variation	Tactile handwritten digits	Recognition baselines	Provides recognition labels; no robot action or grasp-success target.
Macdonald et al. [19] STDP and 3-NN	Repeated tapping and sliding	Optical tactile edge contacts	Edge orientation classification	Specifies sensory learning and readout; task is perception.

Table 2. Continued.

Study and Mechanism	Variation Considered	Data or Case	Evaluation	Scope Relevant Here
Taunyazov et al. [17] Visual–tactile SNN	Robot perception trials	Containers and rotational slip	Task classification	Robot-facing sensing with explicit tasks; differs from force regulation.
Davies et al. [15] Programmable spiking chip	Workload and operating point	Fabricated Loihi	Processor features and task costs	Platform measurements apply to their own workload and boundary.
Frenkel et al. [16] ODIN with SDSP	Digital model and operating point	Fabricated processor	Online learning and chip metrics	Demonstrates SDSP, not the pair-based STDP implemented here.
Yik et al. [7] Benchmark framework	Task and implementation differences	Algorithm and system tracks	Correctness and complexity	Defines measurement architecture rather than a tactile controller.
Present architecture Delta coding and local STDP	Bounded noise and declared grid	Generated taxel and force streams	Conformance, tracking and cost	Executed modular simulation; learned representation is separate from feedback.

2.4 | Positioning of the Present Study

Send-on-delta reporting provides the direct antecedent for the reconstruction mechanism used here [31]. Event-triggered control also has an established stability and scheduling literature, including Tabuada's [32] analysis of stabilizing control tasks. Our controller executes periodically; only its sensory payload is event-coded. Its scalar error bound is a specialization of a standard stable-recurrence argument, with the quantization error made explicit. Likewise, the neuron and STDP primitives are established constructions. The present contribution lies in aligning their assumptions, providing equivalence and conformance checks, and deriving the workload conditions under which the specified implementation choices are useful.

Table 3 identifies the integration questions answered within this scope. Together, these questions move the evaluation beyond an informal pipeline: reconstruction has an error tolerance, learning has an executable update order, and the cost model has a workload and a boundary. The result is a reference architecture that can be challenged or extended by replacing one declared component at a time.

Table 3. Integration questions addressed by the reference architecture.

Question	Relevant Prior Foundation	Resolution in This Study
How does an amplitude event affect feedback error?	Delta reporting [31] and stabilizing control scheduling [32]	A reconstruction bound and a conditional contact-force tracking envelope in Section 3.
When can neural idle ticks be skipped exactly?	Neuron dynamics [9] and explicit plasticity semantics [24]	A model-specific equivalence proof and 150 paired workload checks.
When does gating improve on an already sparse baseline?	Co-design [6] and workload-aware benchmarking [7].	Matched execution policies, full local-update counters and a wake-up break-even condition.
Can average traffic size the input service path?	Event-based sensing and temporal processing [4], [22]	A declared finite-service queue and burst-loss analysis on the generated workload grid.

3 | Architecture and Mathematical Specification

3.1 | Signal Interfaces and Event Reconstruction

Fig. 1 shows the two processing branches and the common accounting boundary. A calibrated aggregate-force channel represents normal contact force in newtons. Spatial taxel channels represent normalized local tactile activity. They use the same encoder construction with independently selected increments, denoted by δ_F and δ_T. This independence matters because reducing force reconstruction error and preserving sensory spike activity need not select the same threshold.

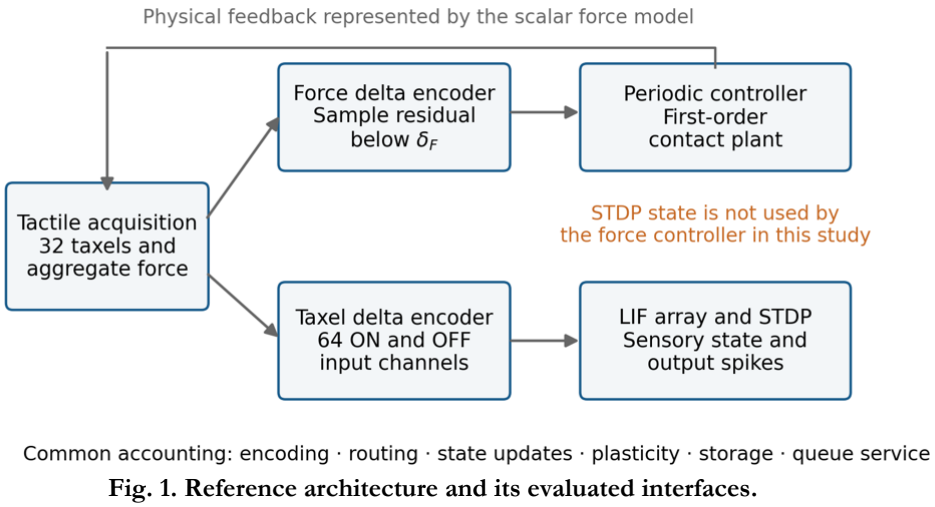


Fig. 1. Reference architecture and its evaluated interfaces.

The aggregate-force feedback and adaptive tactile-array branches are evaluated separately. The figure specifies a computational design, rather than a fabricated circuit.

For the force channel, let $F[k]$ be the physical model state, $y[k]$ the sampled measurement, and $\xi[k]$ bounded measurement noise. The acquisition period is $\Delta=1$ ms. We assume

$$y[k]=F[k]+\xi[k], \qquad |\xi[k]|\le\nu. \tag{1}$$

Let $\hat{y}[k-1]$ be the encoder's previous reconstructed value. A sample can cross several increments in one acquisition interval. We therefore transmit a signed integer multiplicity rather than discard all but the first crossing:

$$m[k]=\operatorname{trunc}\left(\frac{y[k]-\hat{y}[k-1]}{\delta}\right). \tag{2}$$

$$\hat{y}[k]=\hat{y}[k-1]+\delta m[k], \qquad \delta>0$$

Here truncation is toward zero. A nonzero multiplicity can be stored as one timestamped packet, or expanded to $|m[k]|$ unit events of the indicated polarity. We report both packet and unit-event counts. The encoder processes the complete multiplicity at each sample; a capped event output would require a different reconstruction analysis. Initial reconstructed values are explicitly initialized with the trial's calibrated starting condition.

Proposition 1. In exact arithmetic, after processing each sample, the reconstruction obeys

$$|y[k]-\hat{y}[k]|<\delta, \qquad |F[k]-\hat{y}[k]|<\delta+\nu. \tag{3}$$

Proof: Put $s=(y[k]-\hat{y}[k-1])/\delta$. Truncation toward zero leaves $|s-\operatorname{trunc}(s)|<1$. Substituting the reconstruction update gives the first inequality. The second follows from the measurement equation and the triangle

inequality. The argument does not depend on the change in amplitude between samples, provided the full integer multiplicity is retained.

This bound applies at processed sample instants with lossless, ordered delivery. A periodic acquisition front end still performs its measurement and comparison when the output multiplicity is zero. Timestamp quantization, missing events and transport delay must therefore be accounted for separately from the ideal reconstruction property.

3.2 | Neuron State and Causal Local Plasticity

The sensory array contains N input channels and M output neurons, with $N=64$ and $M=16$ in the reference experiment. The first 32 channels encode positive taxel increments and the remaining 32 encode negative increments. Both polarities drive nonnegative synapses; the channel identity preserves the polarity. Let $n_i[k]$ be an integer input-event count, $w_{ji}[k]$ a synaptic weight and $v_j[k]$ the membrane state after reset. A zero-rest, current-based LIF model is

$$v_j^*[k]=\alpha v_j[k-1]+\sum_{i=1}^{N} w_{ji}[k-1]n_i[k],$$
$$z_j[k]=1\{v_j^*[k]\geq\theta\}, \qquad v_j[k]=\big(1-z_j[k]\big)v_j^*[k]. \tag{4}$$

The leak factor is $\alpha=\exp(-\Delta/\tau_m)$, with $\tau_m=20$ ms and threshold $\theta=1$. A neuron emits at most one spike per acquisition bin and resets to zero. It has no bias current, recurrent input, adaptive threshold or refractory state. These choices define a tractable reference model; they are also the assumptions that make the lazy execution result below possible.

Presynaptic and postsynaptic traces summarize strictly earlier events. With decay factors $\beta_+=\exp(-\Delta/\tau_+)$ and $\beta_-=\exp(-\Delta/\tau_-)$, we compute

$$p_i^-[k]=\beta_+ p_i[k-1], \qquad q_j^-[k]=\beta_- q_j[k-1],$$
$$p_i[k]=p_i^-[k]+n_i[k], \qquad q_j[k]=q_j^-[k]+z_j[k]. \tag{5}$$

The traces used for learning are the decayed values before the current spikes are added. This convention excludes pair contributions at zero time difference. Plasticity is applied after current-bin spikes have been determined from the previous weights:

$$\Delta w_{ji}[k]=\eta\big(A_+ z_j[k]p_i^-[k]-A_- q_j^-[k]n_i[k]\big),$$
$$w_{ji}[k]=\mathrm{clip}_{[0,w_{max}]}\big(w_{ji}[k-1]+\Delta w_{ji}[k]\big). \tag{6}$$

The implementation uses $\eta=0.002$, $A_+=1$, $A_-=1.05$, $\tau_+=\tau_-=20$ ms and $w_{max}=0.2$. The 20 ms value is a decay time constant, rather than a hard pairing-window cutoff. We initialize weights independently and uniformly on $[0.02,0.08]$ and reset all traces at the start of each trial. Simultaneous positive and negative contributions are summed before a single clipping operation. Sequential clipping after each pair is a different update rule.

Before clipping, summing the trace increments implements all-to-all pair contributions with the kernel

$$K(\Delta t)=\begin{cases} A_+\exp(-\Delta t/\tau_+), & \Delta t>0,\\ -A_-\exp(\Delta t/\tau_-), & \Delta t<0,\\ 0, & \Delta t=0. \end{cases} \tag{7}$$

This relation identifies the pairing semantics; it does not equate a clipped trajectory with an unconstrained sum of all historical pairs. Projection immediately yields $0\leq w_{ji}[k]\leq w_{max}$ by induction. Bounded weights control the numerical range of the learning state, but do not establish class selectivity, convergence of the weights or improved manipulation. Those properties require additional objectives and experiments.

3.3 | Exact Lazy Execution Under Idle Input

Between active input bins, all membrane states and traces decay. If the previous processed bin was t and the next active bin is k, their pre-input values can be recovered directly:

$$v_j^-[k]=\alpha^{k-t}v_j[t].$$

$$p_i^-[k]=\beta_+^{k-t}p_i[t], \qquad q_j^-[k]=\beta_-^{k-t}q_j[t]. \tag{8}$$

Proposition 2. For the neuron and plasticity model in Section 3.2, processing only active input bins with the preceding decay rule produces the same spikes, weights and reconstructed state at every bin as clocked execution in exact arithmetic. A final decay is required when the last trial bin is idle.

Proof: Immediately after each processed bin, the membrane state is nonnegative and strictly below threshold. If no input arrives, multiplication by $0<\alpha<1$ cannot cross the threshold. There is consequently no output spike and no weight update during an idle interval. Repeated membrane decay equals multiplication by α^{k-t}, and the same semigroup identity holds for both traces. At the next active bin the two implementations therefore have identical pre-input states and previous weights. They perform the same current accumulation, threshold test, reset and clipped update. Induction over active bins proves the result. Applying the geometric decay to a trailing idle interval aligns the terminal states.

This equivalence depends on the specified dynamics. A delayed synaptic current, autonomous oscillation, bias current or time-varying threshold could generate a spike without a new input packet. Such a model requires scheduled internal events and cannot use this particular skipping rule unchanged. Floating-point implementations also replace algebraic equality with a numerical conformance tolerance.

Algorithm 1: At an active bin, first obtain the elapsed tick count and decay the membrane and both traces. Next accumulate the weighted input counts using the previous weights, determine all output spikes and reset their membrane states. Form potentiation and depression from the pre-increment traces, add their contributions and clip each affected weight once. Finally add the current input and output counts to the traces, record the processed timestamp, and emit the output packet set. Idle bins leave the stored weights unchanged; their state is reconstructed by elapsed-time decay when required. This ordering is implemented in the released reference code.

3.4 | Force Feedback and the Reconstruction Budget

The aggregate-force interface is evaluated with a scalar first-order contact model. It represents a stable force response with unit static gain and an additive disturbance, rather than a complete grasping or rigid-body simulator:

$$F[k+1]=aF[k]+(1-a)(u[k]+d[k]), \qquad a=\exp(-\Delta/\tau_c). \tag{9}$$

Here $\tau_c=80$ ms, $u[k]$ is a force command and $d[k]$ is a disturbance in newtons. The sampled proportional controller with reference feedforward is

$$u[k]=\mathrm{clip}_{[0,U]}\big(r[k]+K_c(r[k]-\hat{y}[k])\big). \tag{10}$$

We use gain $K_c=2$ and command limit $U=10$ N. The controller runs at every acquisition tick, including ticks with no new event. Its computational cost is therefore periodic. The periodic-feedback comparator replaces $\hat{y}[k]$ with the current measured force $y[k]$ and otherwise uses the same gain, command limit and plant. An open-loop reference uses $u[k]=r[k]$. The adaptive tactile array supplies neither a hidden disturbance estimate nor a learned command to these controllers.

To expose the control consequence of event reconstruction, define $e[k]=F[k]-r[k]$, $\varepsilon[k]=F[k]-\hat{y}[k]$, and $\Delta r[k]=r[k+1]-r[k]$. During unsaturated, zero-delay operation, substitution gives

$$e[k+1]=(a-(1-a)K_c)e[k]+(1-a)(K_c\varepsilon[k]+d[k])-\Delta r[k]. \quad (11)$$

Proposition 3. Suppose the command remains unsaturated, event delivery is lossless with zero additional delay, $|d[k]|\le D$, $|\Delta r[k]|\le R$, and $\rho=|a-(1-a)K_c|<1$. If $|\varepsilon[k]|\le\delta_F+\nu$, then

$$|e[k]|\le\rho^k|e[0]|+\frac{1-\rho^k}{1-\rho}\,b. \quad (12)$$

$$b=(1-a)(K_c(\delta_F+\nu)+D)+R.$$

Proof: Taking the absolute value of the error recurrence gives $|e[k+1]|\le\rho|e[k]|+b$. Iteration yields a finite geometric series. Its sum is $(1-\rho^k)/(1-\rho)$, which proves the bound. For a constant reference, $R=0$ and the limiting bound is $b/(1-\rho)$.

The bound is deliberately a worst-case statement. It need not order the mean squared errors obtained with different increments under a particular disturbance. For the reference gain and time constant, $\rho=0.9627334$. With $\delta_F=0.1$ N, $\nu=0.02$ N, $D=0.6$ N and a constant reference, the limiting bound is 0.28 N. A separate adversarial recurrence check in the code verifies this example. The force experiment below uses a disturbance bounded conservatively by 0.9 N and contains reference steps; those settings require their own transient and disturbance terms.

A desired constant-reference error envelope E_{max} can be translated into an encoder budget whenever the resulting right-hand side is positive:

$$\delta_F\le\frac{(1-\rho)E_{max}/(1-a)-D}{K_c}-\nu. \quad (13)$$

This condition is sufficient under *Proposition 3*, not an optimal threshold-selection rule. If the right-hand side is nonpositive, the chosen controller and disturbance budget cannot certify the requested envelope by refining the encoder alone. The command limit must also be checked; clipping changes the linear recurrence.

Transport delay contributes a different error. If the force change is bounded by $|F[k+1]-F[k]|\le L\Delta$ and the controller uses a reconstruction delayed by l ticks, then

$$|F[k]-\hat{y}[k-l]|<\delta_F+\nu+Ll\Delta. \quad (14)$$

The inequality follows by summing intervening force changes and applying *Proposition 1* at the delayed sample. It illustrates why a fine amplitude increment cannot compensate for an arbitrarily stale message. The numerical delay study evaluates this effect directly; it does not assert that every delayed configuration inherits the zero-delay bound without the additional rate assumption.

4 | Resource and Timing Model

4.1 | Workload Counters and Storage

We compare execution policies for the same neural trajectory. Dense clocked execution evaluates every input–neuron product at every bin. Sparse clocked execution evaluates only effective input accumulations while retaining periodic membrane and trace updates. Event-gated execution additionally skips idle state updates using *Proposition 2*. Plasticity is event-local in all three policies, so an unnecessarily dense learning baseline does not create the comparison advantage.

Let T be the number of acquisition bins, $S=\sum_{k,i}n_i[k]$ the number of unit input events, and T_a the number of bins with at least one event. The synaptic counts are

$$C_{dense}=NMT,\qquad C_{acc}=MS,\qquad g=\frac{T_a}{T}. \quad (15)$$

A dense multiply–accumulate and an effective synaptic accumulation are separate primitives. Their coefficients can therefore differ. At bin k, let a_k be the number of active input channels and b_k the number of output spikes. For the specified two-term local update, the number of plasticity-term evaluations and the union of potentially written weights are

$$C_{term} = \sum_k (Ma_k + Nb_k).$$

$$C_{write} = \sum_k (Ma_k + Nb_k - a_k b_k). \quad \textbf{(16)}$$

The subtraction avoids counting an overlapping row–column weight twice when the bin's update is combined before clipping. These counters describe scheduled work, including zero-valued changes at touched weights. A design that suppresses zero-trace accesses can reduce them further. For packed multiplicities, the synaptic count MS is conservative relative to one multiply by a packet count; we retain the unit-event convention throughout the arithmetic and queue comparisons.

Reference code uses vectorized floating-point arrays to implement the recurrence. Its Python instruction count is not a hardware operation trace. The counters instead evaluate the explicitly stated execution policies on the generated event and spike streams. In particular, a vectorized reference assignment can physically touch more host memory than the sparse implementation budget counts. This distinction allows functional verification without presenting host execution as a silicon measurement.

For a dense synaptic array, an illustrative storage budget with b_w bits per weight, b_s bits per state, b_t bits for an elapsed-time stamp and a queue of Q entries of b_q bits is

$$B_{store} = NMb_w + (N+2M)b_s + b_t + Qb_q. \quad \textbf{(17)}$$

The state term includes one presynaptic trace per input, one postsynaptic trace per neuron, and one membrane state per neuron. The event-gating policy decays all such states together and uses one common processed timestamp. *Table 4* gives storage examples for 8-bit weights, 16-bit states, a 16-bit timestamp and a conservative 256-entry, 32-bit queue. These are capacity calculations, not a validated fixed-point implementation or an area estimate. The numerical experiments use 64-bit floating point.

Table 4. Illustrative dense-array storage budgets excluding routing tables and implementation overhead.

Taxels	Inputs N	Neurons M	Weights	State Bytes	Total KiB
32	64	16	1,024	192	2.189
100	200	128	25,600	912	26.893
256	512	256	131,072	2048	131.002

4.2 | Energy Boundary and Event-Gating Break-Even

The modeled energy boundary includes periodic taxel acquisition and comparison, input-event handling, synaptic arithmetic, neuron and trace updates, local plasticity, weight writes and a declared common idle power. It excludes motor power, the physical sensor's analog front end and an external host. Those costs can be added when a target platform supplies their values. We do not assign the modeled neural energy to a complete robot.

For policy p, let $C_{p,h}$ be its count of primitive h and e_h the assumed energy per primitive. With physical trial duration $T\Delta$, the model is

$$E_p = P_0 T\Delta + \sum_h e_h C_{p,h} + \mathbf{1}\{p=\text{event}\} e_{wake} T_a. \quad \textbf{(18)}$$

Table 5 reports every coefficient used in the illustrative numerical calculation. The coefficients are assumptions chosen to expose sensitivity and are not fitted to a device or copied from a different processor. Acquisition and local plasticity appear in all policies. A platform comparison would require synthesis or measurement of each primitive and an agreed boundary for the sensing, control and communication paths.

Table 5. Assumed primitive coefficients for sensitivity analysis rather than chip measurements.

Primitive	Energy pJ	Counting Convention
Sample and delta comparison	5.0	Every taxel at every acquisition tick
Input event handling	2.0	Per unit input event
Effective synaptic accumulation	0.9	Per nonzero unit event and fan-out synapse
Dense multiply–accumulate	3.1	Every input–neuron pair and tick
Membrane update	1.0	Per scheduled neuron-state update
Trace decay or update	0.5	Per scheduled trace-state update
Plasticity term	3.0	Per scheduled local term evaluation
Weight write	1.0	Per member of the affected row–column union
Wake-up overhead	100.0	Per active tick in event-gated execution

The sparse clocked and event-gated policies share the same effective synaptic operations and plasticity counts. Their modeled energy difference therefore has a particularly simple form. Put $H=Me_v+(N+M)e_t$, the cost of one complete neuron-and-trace update tick. Then

$$E_{event}-E_{sparse}=T(ge_{wake}-(1-g)H). \tag{19}$$

Proposition 4. With identical common costs and positive H and e_{wake}, event gating uses less modeled energy than sparse clocked execution exactly when

$$g<\frac{H}{H+e_{wake}}. \tag{20}$$

Proof: Rearranging the preceding energy difference gives $g(H+e_{wake})<H$. Both coefficients are positive, so division preserves the inequality. Equality defines the break-even point. □

This result separates activity sparsity from the cost of exploiting it. In the reference array, $H=56$ pJ under *Table 5*. A 100 pJ wake-up charge requires $g<0.35897$; zero wake-up cost makes any nonzero idle interval useful in this model. If two implementations have different static powers, their idle-power difference must be added before applying the comparison. Likewise, the cost of evaluating the elapsed-time decay, including lookup or exponentiation support, must fit within the assumed state-update coefficient or be added as a separate primitive.

4.3 | Burst Service and Deadline Accounting

Average event rate does not bound contact-burst backlog. For a diagnostic unit-event queue, let $A_k=\sum_i n_i[k]$ be the arrivals during bin k, B the service capacity per bin and Q_{max} the post-service storage capacity. We use arrival, service, then capacity enforcement:

$$Q_{k+1}=\min\left(Q_{max},\max(0,Q_k+A_k-B)\right). \tag{21}$$

The corresponding number of dropped unit events in that bin is

$$D_k=\max(0,Q_k+A_k-B-Q_{max}). \tag{22}$$

The buffer model expands packet multiplicities into unit events. A packed-packet implementation can have a different service demand; it should be assessed with its own arrival sequence. In the storage example, a 32-bit

packet can hold a 6-bit channel identity, 8-bit multiplicity, 16-bit tick stamp and 2 flag bits. Trial durations are shorter than the 65.536 s timestamp wrap interval; longer deployments require explicit modular-age or epoch handling.

Queue service is evaluated as a separate transport diagnostic on the tactile workloads. Dropped or delayed packets are not fed back into the neural-equivalence or force-tracking experiments. Consequently, those experiments establish the ideal interface behavior, and the queue study identifies when a particular service design would violate that interface. After event loss, exact delta reconstruction requires recovery of the missing increment or a new absolute reference. Continuing as though the ideal error bound still held would be incorrect.

5 | Reproducible Experimental Protocol

5.1 | Constructed Tactile Workloads

We generated a controlled spatial tactile stream to exercise smooth motion, load changes, oscillation and bounded sensor noise. Taxel positions x_i are equally spaced on [-1,1] for 32 taxels. At time $t=k\Delta$, the normalized signal is

$$s_i(t)=a_T(t)\exp\left[-\left(\frac{x_i-c(t)}{0.35}\right)^2\right]+0.03\sin(18\pi t+3x_i)+\epsilon_i(t). \quad (23)$$

The center is $c(t)=0.7\sin(1.2\pi t)$ and the envelope is $a_T(t)=0.3+0.41\{t\geq 0.5\}-0.251\{t\geq 2.5\}$. Independent uniform noise lies in $[-\nu_T,\nu_T]$. Every trial lasts 4 s. The first sample initializes the reconstructed taxel values; subsequent changes generate events. The signal is a constructed stress workload, rather than a calibrated tactile device model or an observed manipulation sequence.

We crossed five increments, $\delta_T\in\{0.01,0.025,0.05,0.1,0.2\}$, with three noise bounds, $\nu_T\in\{0,0.002,0.01\}$, and ten seeds numbered 0–9. Each of the 150 workloads was evaluated by clocked and lazy execution with the same initial weights. Weight initialization used a separate generator initialized with seed $10000+s$ for workload seed s. We compared all output spikes, final weights, membrane states and traces. The released conformance test also compares the unclipped trace update with an independently enumerated spike-pair sum containing repeated and simultaneous events.

The local learner starts afresh for each trial. No class labels, train–test partition or classification accuracy is involved. A frozen-weight execution is retained for the representative event stream as a diagnostic artifact, while the main equivalence comparison holds learning fixed across the two execution policies. We selected $\delta_T=0.05$, $\nu_T=0.002$ and seed 0 for the plotted raster and weight matrices. The complete grid, rather than the representative trace, supplies the sensitivity results.

5.2 | Contact-Force Scenarios and Comparators

Force trials last 6 s and initialize $F[0]=\hat{y}[-1]=2$ N. The target changes from 2 N to 4 N at 1 s and to 3 N at 4 s. The disturbance is

$$r(t)=2+2\mathbb{1}\{t\geq 1\}-1\mathbb{1}\{t\geq 4\}.$$
$$d(t)=0.3\sin(1.4\pi t)-0.61\{2\leq t<2.3\}. \quad (24)$$

We crossed $\delta_F\in\{0.01,0.025,0.05,0.1,0.2\}$ N, $\nu\in\{0,0.005,0.02\}$ N, delay $l\in\{0,2,10\}$ ticks and 30 noise seeds, numbered 0–29. Each configuration was run with periodic feedback, event feedback and the open-loop reference, yielding 4,050 runs. The same bounded noise sequence was paired across policies. The initialized 2 N reconstruction is held until the delayed first sample can arrive. Acquisition, reference timing, plant dynamics and command limits are common across policies.

Some controls are intentionally repeated across increment settings to preserve the paired design: periodic feedback does not use the event increment, and the open-loop plant ignores sensor noise. Noiseless repeats are also deterministic. The run count therefore describes the executed grid, rather than 4,050 independent physical observations. Hyperparameters were fixed for the reference study; we did not select a winning gain separately for each policy.

5.3 | Metrics and Reproducibility Checks

For an index set I of acquisition bins, the force metric is the Root-Mean-Square Error (RMSE),

$$\mathrm{RMSE}_I = \sqrt{\frac{1}{|I|}\sum_{k\in I}(F[k]-r[k])^2}\,. \tag{25}$$

Whole-trial RMSE includes the reference-step transients. The steady-window metric excludes the first 250 ms and the 250 ms immediately following each reference change; it retains the disturbance pulse. We report mean and sample standard deviation across seeds, along with packet and unit-event counts. For the stated representative force setting, a paired percentile bootstrap resamples the 30 seed-level RMSE differences 10,000 times with generator seed 271828. Acquisition bins are not treated as independent experimental replicates.

The software checks the reconstruction residual at every sample, the synaptic bounds, spike equality and final-state agreement. Scalar force recurrences are cross-checked against the vectorized trial implementation. Raw Comma-Separated Value (CSV) files retain all configurations and outcomes, including saturation counts; no command saturated in the reported grid. *Table 6* records the main simulation and accounting settings. The code uses NumPy and contains no dependency on MATLAB, Brian2, a robot simulator or a proprietary hardware toolchain.

Table 6. Reference experiment settings and evaluation units.

Quantity	Setting	Role
Acquisition period	$\Delta=1$ ms	Both neural and force models
Neural array	$N=64$, $M=16$	32 taxels with two polarity channels
Membrane and trace time constants	20 ms	Exponential decay without a hard pairing cutoff
Threshold and reset	$\theta=1$, reset to 0	At most one output spike per bin
Local update	$\eta=0.002$, $A_+=1$, $A_-=1.05$	Strictly earlier traces; simultaneous pairs excluded
Weights	Initial $[0.02,0.08]$; allowed $[0,0.2]$	Independent initialization each trial; 64-bit reference
Tactile grid	5 increments × 3 noise bounds × 10 seeds	150 workloads, each with two execution policies
Force plant and gain	$\tau_c=80$ ms; $K_c=2$; $U=10$ N	Identical dynamics and command limits across policies
Force grid	5 increments × 3 noise bounds × 3 delays × 30 seeds	1,350 configurations with three control policies
Trial durations	Tactile 4 s; force 6 s	No external recognition dataset used
Transport diagnostic	256 unit events; service 8–128 events/ms	Five capacities for every tactile workload
Numerical tolerance	10^{-10}	Terminal neural-state agreement; exact spike equality
Illustrative idle power	10 μW	Common overhead within the modeled boundary

6 | Results

6.1 | Reconstruction and Force Regulation

Fig. 2 shows the representative force response with δ_F=0.05 N, ν=0.005 N, zero transport delay and seed 0. The reconstructed force follows the physical state within the combined encoding and noise envelope at acquisition instants. The feedback policies reject much of the constructed disturbance relative to open loop, while both exhibit transients after target changes. The figure reports the generated force and event sequence; it is not a grasping trial.

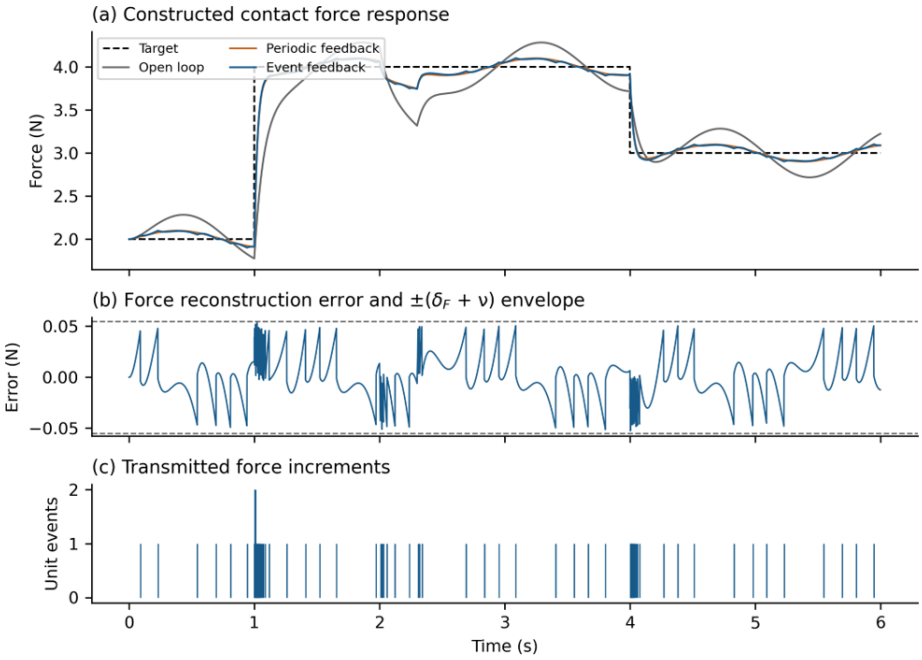


Fig. 2. Executed force-control example with a 0.05 N force increment and 0.005 N noise bound.

The reconstruction envelope is evaluated at processed acquisition instants.

Table 7 quantifies the increment sweep at the same noise level and zero delay. At δ_F=0.05 N, event feedback achieved a steady-window RMSE of 0.082267 N with a sample standard deviation of 0.000136 N. Periodic feedback achieved 0.083041 N, and open loop achieved 0.232488 N. The mean paired event-minus-periodic difference was −0.000774 N, with a 95% bootstrap interval of [−0.000824, −0.000723] N. This narrow interval reflects repeated noise realizations for the same plant and disturbance. It does not quantify uncertainty across objects, contact mechanics or robots.

Table 7. Event-feedback results at a 0.005 N noise bound and zero delay over 30 paired seeds.

Increment N	Steady RMSE N	Sample SD N	Mean Packets	Whole-trial RMSE N
0.01	0.083118	0.000066	331.6	0.137569
0.025	0.083433	0.000084	173.9	0.137687
0.05	0.082267	0.000136	94.9	0.136407
0.1	0.078894	0.000230	51.0	0.133058
0.2	0.121506	0.000209	44.0	0.160131

Event feedback transmitted 94.9 packets on average per 6,000 acquisition ticks at the 0.05 N setting, a 98.42% reduction relative to transmitting one force value every tick. It still acquired 6,000 measurements and executed 6,000 control updates. The reduction is consequently a payload-transmission result. It is not a measured energy reduction and does not imply that the controller itself became event-triggered.

The RMSE curve is nonmonotonic over the smaller increments (see *Fig. 3*). With a 0.1 N increment, event feedback had a lower steady-window RMSE for this disturbance than at 0.01 N; at 0.2 N it worsened to 0.121506 N. Quantization changes when corrective action is applied and can alter a particular disturbance response. *Proposition 3* bounds the worst-case effect of reconstruction error and does not predict this mean-error ordering. The small improvement over periodic feedback in the selected scenario should therefore be interpreted as a property of that scenario, rather than a general control advantage of event encoding.

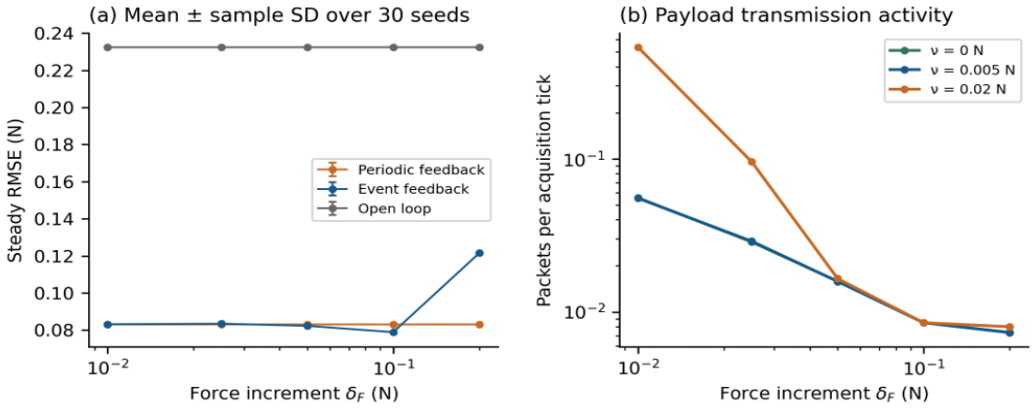


Fig. 3. Force tracking and communication across the declared increment grid. Error bars show seed-level standard deviations.

The transmission plot counts packets, including packed multiple increments. Additional delay increased the steady-window RMSE at the representative setting from 0.082267 N at zero delay to 0.082659 N at 2 ms and 0.084303 N at 10 ms. The raw output records the full delay–noise–increment grid. These values show why amplitude accuracy and message freshness need separate budgets, even in a stable scalar plant.

6.2 | Neural Conformance and Learning Activity

Across all 150 tactile workloads, clocked and lazy execution produced identical output spikes. The maximum absolute discrepancy over terminal weights, membrane states and traces was 1.3323×10^{-15}, below the declared 10^{-10} numerical tolerance. The independent direct-pair and trace checks differed by 6.07×10^{-18} for the unclipped test sequence. These results support the mathematical equivalence under the reference assumptions and detect errors that would be hidden by comparing only a final classification score. *Fig. 4* shows generated input events, output spikes and the initial and final synaptic matrices for the representative tactile workload. The weights changed by a mean absolute amount of 0.004995 across ten seeds at this increment and noise level. This is evidence that the implemented rule updates local state. No labels or behavioral objective were supplied, so the matrices are not interpreted as learned object categories or successful motor adaptation.

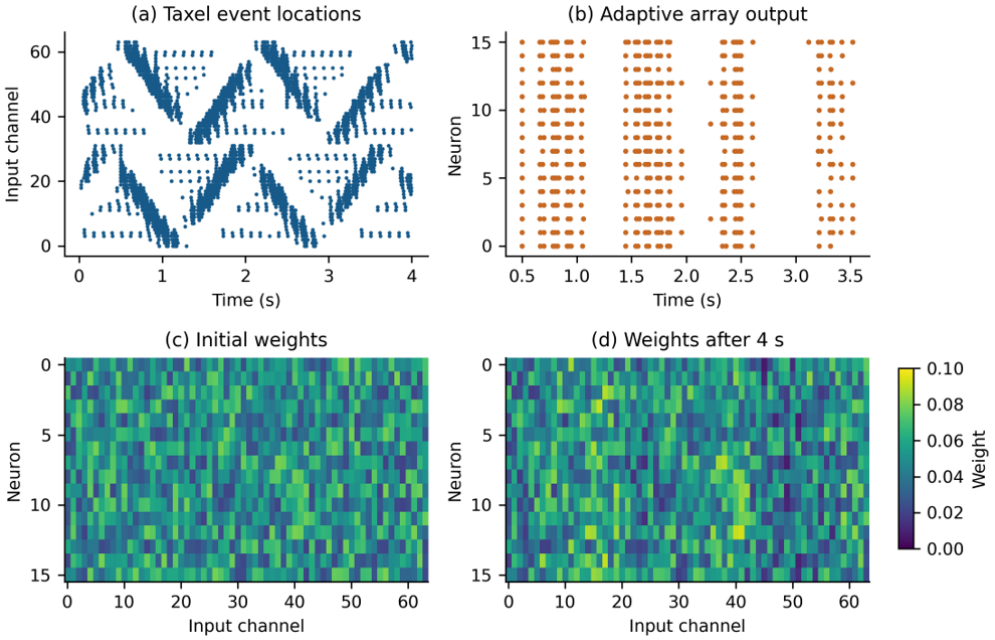


Fig. 4. Generated tactile events and local synaptic adaptation for the representative four-second workload.

The initial and final weight panels share the same color scale.

Table 8 reveals a limit that a sparsity-only analysis would miss. At δ_T=0.2 and ν_T=0.002, the array emitted no output spikes in any of the ten trials and its weights did not change. Increasing the increment had reduced the input drive below the firing requirement. At δ_T=0.1, only 32 output spikes occurred on average, compared with 497.3 at 0.05 and 8,804.1 at 0.01. The most sparse workload is therefore not automatically a useful learning configuration.

Table 8. Neural workload and adaptation at a 0.002 taxel noise bound over ten seeds.

δ_T	Unit Input Events	Active Tick Fraction	Output Spikes	Mean $\lvert\Delta w\rvert$	Weight Writes
0.01	18,264.1	0.973500	8,804.1	0.023741	792,171.8
0.025	6,327.8	0.734675	2,350.8	0.017260	239,801.0
0.05	2,529.4	0.418625	497.3	0.004995	69,416.6
0.1	1,019.4	0.203025	32.0	0.000404	17,657.6
0.2	421.1	0.093250	0.0	0.000000	6,689.6

Noise also eroded sparsity. At δ_T=0.01, the mean number of unit input events rose from 17,842.0 without noise to 18,264.1 at a 0.002 noise bound and 30,134.6 at a 0.01 noise bound. *Fig. 5* connects this traffic to the fraction of active ticks and the output activity. These quantities must be considered together: lowering the increment preserves small tactile changes but increases routing and learning work; raising it saves activity while eventually disabling the local learner.

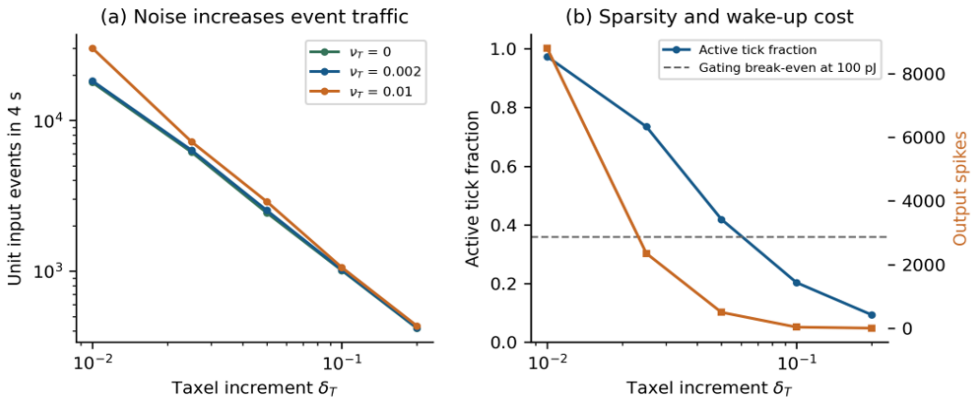


Fig. 5. Event traffic, active ticks and output activity across the tactile increment grid.

The horizontal line is the event-gating break-even threshold under the illustrative 100 pJ wake-up coefficient.

6.3 | Cost Sensitivity and Burst Behavior

At the representative tactile setting, the mean effective synaptic accumulation count was 40,470.4 per trial, compared with 4,096,000 dense input–neuron products. The corresponding learning workload included 71,155.2 plasticity-term evaluations and 69,416.6 scheduled weight writes. These local updates are substantial relative to the effective forward accumulations and must remain in the model when assessing online learning.

Under the illustrative coefficients, dense clocked dynamic energy was 13.8495 µJ over four seconds, sparse clocked energy was 1.1884 µJ, and event-gated energy was 1.2256 µJ. Event gating therefore cost approximately 3.13% more than sparse clocked execution at this setting. Its active fraction, 0.418625, exceeded the break-even fraction 0.35897 for a 100 pJ wake-up charge. At δ_T=0.1, the active fraction fell to 0.203025 and event gating used less modeled energy than sparse clocked execution, although neural activity also fell sharply. The energy choice cannot be separated from the representation requirement.

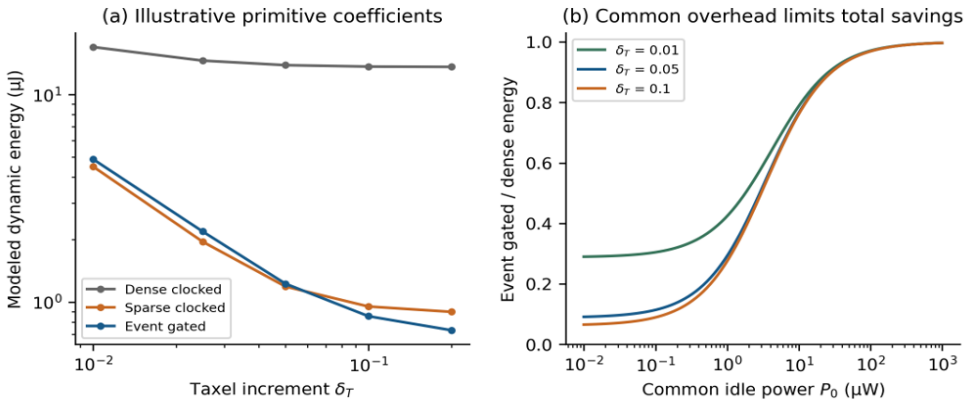


Fig. 6. Illustrative energy calculations and common-overhead sensitivity.

Coefficients are declared assumptions, not device measurements. The total-energy ratio approaches one as common idle power dominates.

Adding a common idle power of 10 µW gave 53.8495 µJ for dense clocked execution and 41.2256 µJ for event gating at the representative setting. The numerical reduction was then 23.44%, substantially smaller than the dynamic-only reduction. *Fig. 6* shows how this ratio changes when the assumed common idle power varies. A larger acquisition, host or actuator contribution would further reduce the fraction of whole-system energy attributable to the saved neural work. Conversely, a hardware encoder that avoids periodic sampling

could change the common acquisition term; that is a different implementation from the sampled front end tested here.

The queue diagnostic exposed burst sensitivity even when the average rate was modest (*Fig. 7*). For δ_T=0.01 and ν_T=0.002, the mean input rate was about 4.57 unit events/ms. A queue serving 8 unit events/ms nevertheless dropped as many as 85 events in a trial because abrupt load changes generated concentrated arrivals. With a 256-event post-service buffer, increasing service to 64 events/ms still left a worst observed loss of 29 events; 128 events/ms removed losses across these ten trials, with a maximum backlog of 221. At δ_T=0.05, service at 64 events/ms left no post-service backlog in the corresponding trials.

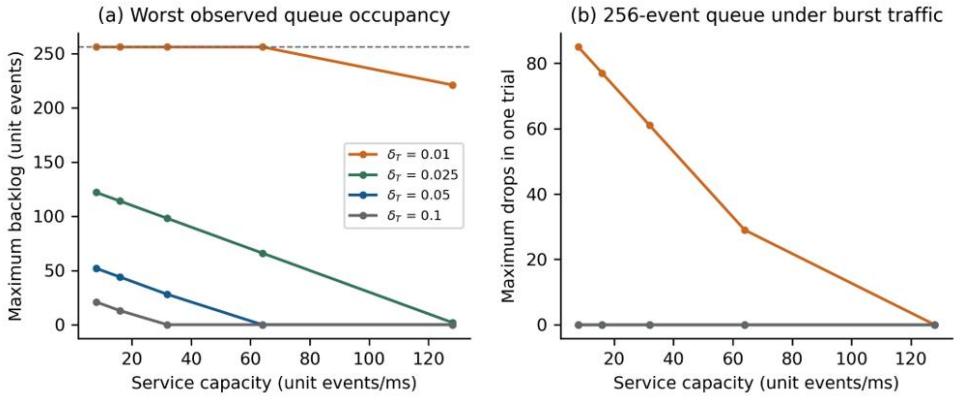


Fig. 7. Unit-event queue sensitivity for the declared transport model. Curves summarize the worst observed backlog and loss across ten seeds at a 0.002 taxel noise bound.

The queue results do not establish a hardware clock frequency. Service is an assumed number of unit events per millisecond, and no register-transfer implementation was synthesized. They provide a concrete capacity requirement for a future design and show that average traffic alone would have selected an inadequate buffer–service combination in the high-resolution case.

7 | Discussion and Implementation Implications

The analysis suggests that tactile neuromorphic design should begin with a representation requirement. The force channel can translate a tracking envelope into an amplitude increment under the assumptions of *Proposition 3*. The sensory branch additionally needs enough input drive and temporal structure to sustain useful neuronal responses. A threshold that nearly eliminates events may improve an operation counter while destroying the learning mechanism. In the reference workload, the disappearance of output spikes at the coarsest increment makes this failure visible without a speculative accuracy claim.

The execution comparison also changes the relevant baseline. A sparse conventional implementation can exploit the same event stream and local update schedule as an event-gated core. Comparing only with dense arithmetic would conceal the wake-up penalty and overstate the architectural benefit. The break-even relation gives an implementation decision rule once the cost of a state update and wake-up can be measured. Its coefficients should include the actual elapsed-time decay method. A lookup-table approximation, multiplication sequence or dedicated decay unit can have different precision and energy behavior from the floating-point reference.

Memory organization must support both forward processing and plasticity. Input events access a synaptic column, while postsynaptic spikes access a row. A simple column-oriented store can therefore make the depression path efficient while increasing the cost of potentiation, or the reverse for row-oriented storage. Banking, a transpose index, time multiplexing or an alternative local rule can address this tension. Each choice

changes memory traffic, storage and scheduling. The logical write-union count is a useful lower-level specification, but it does not eliminate banking conflicts or prescribe a physical layout.

A digital prototype would first establish integer arithmetic, reset behavior, saturation and timestamp rollover against the reference trajectory. The 8-bit storage examples do not yet demonstrate that the learned dynamics survive 8-bit weight quantization. A mixed-signal design introduces additional variability and peripheral requirements, as discussed in the device literature [27], [28]. The reference architecture provides the equations and workload needed for those comparisons, while a circuit implementation must supply its own precision, timing and energy evidence.

The force controller supplies a clear behavioral interface but a deliberately narrow plant. Real manipulation includes changing contact geometry, tangential force, friction, actuator dynamics, saturation and object motion. The scalar model does not predict grasp success or dexterity. A subsequent learned-control study would need a labeled or reward-defined readout, a defined action space and repeated closed-loop trials. Local STDP can remain a sensory adaptation component in that study, but its causal contribution would have to be isolated through frozen-weight and matched task-learning baselines.

Public tactile datasets provide complementary steps toward that goal. ST-MNIST and STEMNIST can test recognition under participant-aware partitions; edge-orientation and event-driven visual–tactile tasks can test representations closer to contact geometry and slip [17–20]. Their use should preserve sample provenance, prevent participant or trial leakage and report task-appropriate uncertainty. For an embodied claim, the experiment must ultimately connect predictions to actual actions and interaction outcomes, consistent with current physical benchmarking work [8].

Our numerical evidence is conditional on generated signals, one neuron family, fixed gains and declared cost coefficients. Seed variation quantifies initialization or bounded measurement-noise effects within that model family. It does not span fabrication variation, unfamiliar objects or realistic contact dynamics. Queue loss was diagnosed separately rather than injected into the ideal neural and control simulations. This modular design makes individual assumptions inspectable, but a fully integrated experiment is required to establish the performance of the complete sensor–learning–control path under transport faults.

8 | Conclusion

We specified and executed a neuromorphic tactile reference architecture in which reconstruction accuracy, local plasticity and execution cost have explicit mathematical roles. The delta encoder preserves an amplitude-error bound when full, ordered increments are delivered. The stated LIF and trace-based STDP model permits exact lazy execution because idle state decay cannot generate a spike. A separate sampled force controller translates reconstruction error into a conditional tracking envelope for a first-order contact model.

The experiments established neural execution conformance and quantified the communication–tracking trade-off under declared scenarios. They also identified operating limits: coarse encoding can silence learning, burst traffic can defeat a queue sized from the average rate, and event gating can exceed the energy of sparse clocked processing when wake-up costs are large. These findings support a design process that evaluates information preservation and implementation overhead together. The released code and outputs provide a reproducible basis for subsequent fixed-point implementation, hardware measurement and task-trained robotic experiments.

Data and Code Availability

The accompanying research package contains the generated event workloads through their deterministic generator, complete run-level CSV outputs, representative state traces, analysis and figure scripts, environment information and bibliography verification records. All reported numerical results can be regenerated without an external dataset. No new human-participant or physical-robot data were collected for this computational study.

Acknowledgments

The authors acknowledge the researchers whose previous work on neuromorphic engineering, tactile sensing, spiking neural networks, and robotic force control provided the theoretical foundation for this study.

Author Contributions

M. L. N.: Conceptualization, methodology, software development, formal analysis, and writing—original draft.

P. D. S.: Mathematical analysis, validation, and technical review.

A. O. O.: Investigation, interpretation of results, and writing—review and editing.

All authors reviewed and approved the final manuscript.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflicts of Interest

The authors declare that they have no conflicts of interest related to the publication of this paper.

References

[1] Mead, C. (1990). Neuromorphic electronic systems. *Proceedings of the IEEE*, *78*(10), 1629–1636. https://doi.org/10.1109/5.58356

[2] Bartolozzi, C., Indiveri, G., & Donati, E. (2022). Embodied neuromorphic intelligence. *Nature Communications*, *13*(1), 1024. https://doi.org/10.1038/s41467-022-28487-2

[3] Liu, F., Deswal, S., Christou, A., Sandamirskaya, Y., Kaboli, M., & Dahiya, R. (2022). Neuro-inspired electronic skin for robots. *Science Robotics*, *7*(67), eabl7344. https://doi.org/10.1126/scirobotics.abl7344

[4] Liu, T., Brayshaw, G., Li, A., Xu, X., & Ward Cherrier, B. (2025). Neuromorphic touch for robotics—A review. *Neuromorphic Computing and Engineering*, *5*(3), 32001. https://doi.org/10.1088/2634-4386/adf091%0A%0A

[5] Roy, K., Jaiswal, A., & Panda, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. *Nature*, *575*(7784), 607–617. https://doi.org/10.1038/s41586-019-1677-2%0A%0A

[6] Frenkel, C., Bol, D., & Indiveri, G. (2023). Bottom-up and top-down approaches for the design of neuromorphic processing systems: Tradeoffs and synergies between natural and artificial intelligence. *Proceedings of the IEEE*, *111*(6), 623–652. https://doi.org/10.1109/JPROC.2023.3273520

[7] Yik, J., den Berghe, K., Den Blanken, D., Bouhadjar, Y., Fabre, M., & Hueber, P. (2025). The neurobench framework for benchmarking neuromorphic computing algorithms and systems. *Nature Communications*, *16*(1), 1545. https://doi.org/10.1038/s41467-025-56739-4

[8] D'Angelo, G., Pedersen, J. E., Hassan, T., Cianchetti, M., Bongard, J., & Iida, F. (2026). A benchmarking framework for embodied neuromorphic agents. *Nature Machine Intelligence*, *8*(3), 300–312. https://doi.org/10.1038/s42256-026-01197-w

[9] Indiveri, G., Linares Barranco, B., Hamilton, T. J., Schaik, A. van, Etienne Cummings, R., & Delbruck, T. (2011). Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, *5*, 73. https://doi.org/10.3389/fnins.2011.00073

[10] Thakur, C. S., Molin, J. L., Cauwenberghs, G., Indiveri, G., Kumar, K., & Qiao, N. (2018). Large-scale neuromorphic spiking array processors: A quest to mimic the brain. *Frontiers in Neuroscience*, *12*, 891. https://doi.org/10.3389/fnins.2018.00891

[11] Bi, G., & Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience*, *18*(24), 10464–10472. https://doi.org/10.1523/JNEUROSCI.

[12] Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, *3*(9), 919–926. https://doi.org/10.1038/78829

[13] Diehl, P. U., & Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, *9*, 99. https://doi.org/10.3389/fncom.2015.00099

[14] Merolla, P. A., Arthur, J. V, Alvarez Icaza, R., Cassidy, A. S., Sawada, J., & Akopyan, F. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, *345*(6197), 668–673. https://doi.org/10.1126/science.1254642

[15] Davies, M., Srinivasa, N., Lin, T. H., Chinya, G., Cao, Y., & Choday, S. H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, *38*(1), 82–99. https://doi.org/10.1109/MM.2018.112130359

[16] Frenkel, C., Lefebvre, M., Legat, J. D., & Bol, D. (2018). A 0.086-mm² 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28-nm CMOS. *IEEE Transactions on Biomedical Circuits and Systems*, *13*(1), 145–158. https://doi.org/10.1109/TBCAS.2018.2880425

[17] Taunyazov, T., Sng, W., See, H. H., Lim, B., Kuan, J., Ansari, A. F., & Soh, H. (2020). *Event-driven visual-tactile sensing and learning for robots*. https://doi.org/10.48550/arXiv.2009.07083

[18] See, H. H., Lim, B., Li, S., Yao, H., Cheng, W., Soh, H., & Tee, B. C. K. (2020). *ST-MNIST--the spiking tactile MNIST neuromorphic dataset*. https://doi.org/10.48550/arXiv.2005.04319

[19] Macdonald, F. L. A., Lepora, N. F., Conradt, J., & Ward Cherrier, B. (2022). Neuromorphic tactile edge orientation classification in an unsupervised spiking neural network. *Sensors*, *22*(18), 6998. https://doi.org/10.3390/s22186998

[20] Tripathi, A., Gaishan, L., Fu, Z., Bartolozzi, C., Shi, B. E., & Basu, A. (2026). *STEMNIST: Spiking tactile extended MNIST neuromorphic dataset*. https://doi.org/10.48550/arXiv.2601.01658

[21] Patil, C. S., Ghode, S. B., Kim, J., Kamble, G. U., Kundale, S. S., & Mannan, A. (2025). Neuromorphic devices for electronic skin applications. *Materials Horizons*, *12*(7), 2045–2088. https://doi.org/10.1039/d4mh01848f

[22] Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., & Censi, A. (2020). Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(1), 154–180. https://doi.org/10.1109/TPAMI.2020.3008413

[23] Aitsam, M., Davies, S., & Di Nuovo, A. (2022). Neuromorphic computing for interactive robotics: A systematic review. *IEEE Access*, *10*, 122261–122279. https://doi.org/10.1109/ACCESS.2022.3219440

[24] Morrison, A., Diesmann, M., & Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, *98*(6), 459–478. https://doi.org/10.1007/s00422-008-0233-1

[25] Neftci, E. O., Mostafa, H., & Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, *36*(6), 51–63. 10.1109/MSP.2019.2931595

[26] Navarro Guerrero, N., Toprak, S., Josifovski, J., & Jamone, L. (2023). Visuo-haptic object perception for robots: an overview. *Autonomous Rrobots*, *47*(4), 377–403. https://doi.org/10.1007/s10514-023-10091-y%0A%0A

[27] Choi, S., Yang, J., & Wang, G. (2020). Emerging memristive artificial synapses and neurons for energy-efficient neuromorphic computing. *Advanced Materials*, *32*(51), 2004659. https://doi.org/10.1002/adma.202004659

[28] Sehgal, A., Dhull, S., Roy, S., & Kaushik, B. K. (2024). Advancements in memory technologies for artificial synapses. *Journal of Materials Chemistry c*, *12*(15), 5274–5298. https://doi.org/10.1039/d3tc04131j

[29] Zhou, S., Gao, C., Delbruck, T., Verhelst, M., & Liu, S.-C. (2025). Exploiting neuro-inspired dynamic sparsity for energy-efficient intelligent perception. *Nature Communications*, *16*(1), 9928. https://doi.org/10.1038/s41467-025-65387-7

[30] Chowdhury, S. S., Sharma, D., Kosta, A., & Roy, K. (2025). Neuromorphic computing for robotic vision: Algorithms to hardware advances. *Communications Engineering*, *4*(1), 152. https://doi.org/10.1038/s44172-025-00492-5

[31] Miskowicz, M. (2006). Send-on-delta concept: An event-based data reporting strategy. *Sensors*, *6*(1), 49–63. https://doi.org/10.3390/s6010049

[32] Tabuada, P. (2007). Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, *52*(9), 1680–1685. https://doi.org/10.1109/TAC.2007.904277