**Institute for Physical AI** @ John Bailey Institute · The Charlot Lab

QUANTUM INFORMATION AT THE EDGE

Technical Report TR-2026-28
Survey / Position · Preprint v0.1
5 August 2026

QUANTUM INFORMATION SCIENCE FOR A BODY ON ITS OWN HARDWARE

# Quantum Information Science for Edge Physical AI

A robot at the edge has a fixed battery and no datacenter, so the question is precise: where does quantum information science help a body compute on its own hardware today? The grounded answer is that most of the near-term value is quantum-*inspired* structure run on classical silicon you can already carry, and this paper measures exactly where it is real, where it is not, and why.

**Charlot Lab**, Institute for Physical AI @ JBI

Draft · 2026-07 · every lab-bench number here is a measurement from the lab's open research record; figures carrying a reference number are quoted from that source under its own protocol and are not re-measured here · companion to the Quantum information at the edge topic

Tensor-network compression · measured

Pure-Rust factored op · runtime-fused · measured

Planning as Ising · solved classically

Open · on-device · no quantum computer required

Quantum information science reaches embodied AI through several doors, and only some of them open at the edge today. This paper walks each with a measurement, not a promise. (1) **Tensor-network compression** (the mathematics of entanglement, borrowed from many-body physics) is the one that ships now: a control policy *trained* in factored form matches the full policy's accuracy at **32× fewer parameters**, and a pure-Rust factored kernel cuts the on-device **multiply** count by the same factor, verified against the dense layer. But the lesson underneath is sharp: you cannot *squeeze* a dense policy after training (even 2× compression keeps only 58% of its accuracy); you must train the compact form. (2) **Planning as Ising / QUBO** (casting inverse kinematics,

trajectory, or waypoint ordering as the binary problem a quantum annealer consumes) is real but offline: classical simulated annealing on the robot's own processor finds the optimum on a small instance, and the durable value is the formulation, not the machine. (3) **Reservoir dynamics** for temporal control live on quantum hardware today, with a classical shadow that runs at the edge. (4) A thread of a different kind: quantum's *own* hardest bottleneck (the real-time error-correction decoder that must read the syndrome every microsecond and correct it before the next cycle) is itself a classical, low-latency, on-device compute problem, the exact layer this lab works in. (5) **Energy-based fusion** is not merely deployable but *deployed*: Boltzmann-machine sensor fusion under GPS spoofing is what a $2M U.S. Army program runs on NVIDIA Jetson edge GPUs today, and our small readable version nearly reaches a not-deployable oracle while beating the outlier-resistant median, until the corrupted sensors are the majority, where we report our own failure. Throughout, the field's bar is **dequantization**: many claimed quantum speedups are matched by classical algorithms, so the discipline is to chase where the *structure* helps a body, not where the word *quantum* sells. None of this requires a quantum computer; it is classical mathematics run on the hardware the robot already carries.

## 1 · What can quantum thinking do for a robot today?

The frontier of quantum computing is measured in qubits and error rates, in datacenters and dilution refrigerators. A robot in the field has none of that: a fixed battery, a modest processor, and a control loop that must close in milliseconds. So the useful question is not "will quantum computers change AI" but the narrow one an engineer can act on: **which ideas from quantum information science help a body compute on its own hardware, today?** The value splits cleanly. The *quantum-inspired* ideas (structures and algorithms discovered in physics but run on classical chips) are deployable now. The quantum *hardware* is a longer-horizon accelerator for specific offline subproblems, and the literature is candid that its advantage, where it exists at all, is narrow and often *dequantizable*.[1] This paper takes the three doors in turn, then a fourth thread where quantum's own bottleneck turns out to be an edge-compute problem, and a fifth where the quantum-inspired pattern is already deployed on robots today, each with a measurement from the lab's open bench.

## 2 · Which thread ships today?

A quantum many-body state has astronomically many amplitudes, yet if its entanglement is limited it can be written compactly as a **tensor network**, a chain of small cores. The same decompositions factor a neural network's weight tensors, and truncating the **bond dimension** $r$ trades size for accuracy.[2] For a robot this is the lever: fit a large policy or world model onto the device it carries. But *how* you apply it decides whether it works. We trained a small MLP to imitate a smooth nonlinear control policy and asked two questions on the same task.

| middle weight, 256×256 = 65,536 params | compression | accuracy kept |
|---|---|---|
| **squeeze the dense policy after training** (rank-64) | 2× | 58% |
| (rank-16) | 8× | 34% |
| (rank-4) | 32× | 0% |
| **train the policy in factored form** (rank-4) | 32× | 100% |
| (rank-2) | 64× | 98% |

The result is measured, and its lesson is not the obvious one. **Squeezing a dense policy after training, with no recovery step, keeps only a fraction of its accuracy on this bench, 58% at 2x, 34% at 8x, 0% at 32x. The binding constraint is algorithmic, not arithmetic: the from-scratch dense weight sits near a high-rank initialization, so the structure the truncation needs is not there to find. Adding a fine-tune pass after truncation is the measured move that shifts that boundary**: a from-scratch dense weight sits near its high-rank random initialization (a rank-16 truncation captures barely a tenth of its energy), so truncating it throws the policy away even at gentle compression. But **training the policy in factored form from the start matches full accuracy at 32× fewer parameters**, because the control function itself is low-complexity, and the compact form has room to fit it. The deployable recipe is *train-compact, not compress-after* (on a pretrained model, truncate then fine-tune to recover), exactly what the tensorization literature reports on large models. You can watch both curves computed live in your browser.

The parameter saving is only half the value; the other half is compute, and it is real on the device. A dense layer $y = Wx$ costs $\text{rows}\times\text{cols}$ multiplies; a factored layer $W = UV$

computes $y = U(Vx)$ in $r(\text{rows}+\text{cols})$ multiplies, never forming the dense matrix. Our pure-Rust kernel measures it:

| layer | rank | dense multiplies | factored multiplies | saving |
|---|---|---|---|---|
| 512×512 | 16 | 262,144 | 16,384 | 16× |
| 1024×1024 | 32 | 1,048,576 | 65,536 | 16× |

The factored output is identical to the dense output to floating-point tolerance, because the weight was exactly rank-$r$. This is now a first-class op in the lab's pure-Rust compute stack, a factored-linear layer (`linear_factored`) that runs on the very same GPU matmul kernel the dense path already dispatches, so the saving is end-to-end and not a standalone demo. Verified against the dense layer on the live backend (identical to floating-point tolerance), it measures a real speedup that climbs with batch width toward the multiply-count ceiling:

| 4096×4096 layer | rank | single-token decode | batch (64 tokens) | ceiling |
|---|---|---|---|---|
| factored vs dense, measured on the live GPU | 128 | 2.2× | 15.7× | 16× |
| factored vs dense, measured on the live GPU | 64 | 1.4× | 17.0× | 32× |

The result is in the two columns. At batch width the layer is compute-bound and the measured speedup nearly reaches the multiply-count ceiling; at single-token decode two kernel dispatches and memory bandwidth, not the multiplies, set the floor, so the win is a smaller constant. The saving is real in both regimes (fewer weights read *and* fewer multiplies, on the classical hardware the robot already carries), but the clock, not the flop count, is what we report.

How far does the chain go? The two-core factorization is the simplest tensor network; the general case is a chain of many cores, and its extreme (the **quantized tensor-train**, which splits each index into bits and factors the weight into $\sim\!\log_2 N$ tiny cores) crushes *structured* weights far beyond low-rank. Measured on a smooth kernel, the eight-core chain reaches 157× compression at $0.08$ reconstruction error, where a two-core low-rank at the same error

manages only $\sim\!32\times$. But on a *random* weight neither helps at all (both leave $\sim\!98\%$ of the matrix behind). That is the through-line once more: the compression is only ever as real as the structure present, which is exactly why the deployable recipe is to **train** the compact form, so the structure is there by construction. (The chain's own caveat is priced too: because it reuses each core across the swept bits, its multiply count beats dense only at small bond, so the on-device kernel stays with the clean two-core case for now.)

## 3 · Can planning be posed as an Ising problem?

Many of a robot's planning subproblems (inverse kinematics, trajectory, posture, waypoint ordering) can be written as a binary quadratic (QUBO / Ising) objective, which is exactly the form a quantum annealer consumes.[3] A recent study mapped inverse kinematics to a QUBO and ran it on a D-Wave annealer, reporting up to 30× on large instances, while stating plainly that it does *not* beat state-of-the-art continuous solvers, that the QUBO grows super-linearly with resolution, and that real-time closed-loop control is out of reach on today's hardware. We took the durable part (the formulation) and solved it where a robot actually can: on its own processor. Casting a 7-waypoint inspection tour as the exact TSP-QUBO and running classical simulated annealing (feasibility-preserving moves on the same Ising objective) hit the brute-force optimum on **8 of 8** runs.

The reading is that the **formulation travels to the edge; the quantum computer, for now, does not**. The Ising objective is an ordinary energy landscape a classical solver descends on the robot itself, and on real-time control it stays classical and continuous. Where a quantum annealer may earn its place is on far larger, offline, combinatorial instances (a scheduling or task-allocation problem solved in the cloud before the robot moves), and even there it must beat strong classical heuristics, which so far it does not.

## 4 · What can reservoir dynamics do for temporal control?

Control is a time-series problem, and quantum reservoir computing maps a temporal signal into rich, high-dimensional quantum dynamics

that give cheap memory, working best at the "edge of chaos."[4] Today it lives on noisy quantum hardware, so it is not an edge method. But it has a classical shadow, ordinary reservoir computing (echo-state networks), which is the same idea run on the processor the robot already has, and that *is* deployable. The quantum version is worth tracking as the hardware matures; the classical version is worth teaching and using now. This thread is the least mature of the three, and we hold it as a watch item rather than a result.

## 5 · Where does the real bottleneck live?

The first three threads ask where quantum *ideas* help a body. This one turns the question around: quantum's *own* hardest bottleneck is a classical, edge-flavored compute problem. Under every hardware bet (Google's superconducting surface codes, IBM's quantum-LDPC codes, Microsoft's topological approach) sits the same unglamorous requirement. A classical program must read the error syndrome every cycle (~1 microsecond for superconducting qubits), decode it, and return a correction before the next cycle, forever, for every logical qubit at once. Google's Willow demonstrated the first below-threshold surface-code memory (logical error suppressed by 2.14× per two units of code distance, down to 0.14% per cycle at distance 7) with an *integrated real-time decoder*;[5] IBM's move to quantum-LDPC codes cuts the physical-to-logical overhead roughly tenfold but leans on a decoder that runs in real time on classical hardware;[6] and an FPGA neural decoder now closes the loop in 550 ns inside a 1.25 $\mu$s cycle.[7] (A cautionary note on reading the headlines: Microsoft's 2025 topological-qubit paper was *not* retracted, as is sometimes reported. A 2021 Majorana paper was; the 2025 one carries an editorial caveat and its reviewers concluded the data do not evidence Majorana zero modes.[8] The distinction matters when you are judging which hardware claim is evidenced.)

We measured the decoder wall on the lab's own bench, using the repetition code as the canonical decoder proxy so every number can be checked against an exact analytic ground truth (the surface code multiplies the graph size by roughly the code distance but the real-time challenge and the near-linear-decoder response are the same shape). The wall has three faces:

| face of the wall | measured | meaning |
|---|---|---|
| naive lookup-table decoding | $2^{d-1}$ | a distance-25 code needs 16.7M table entries; tables don't scale, so machines use algorithmic decoders |
| a correct decoder, per instance | 2 ns → 50 ns | distance 7 → 1001, near-linear, all under 5% of the 1 µs cycle budget for one logical qubit |
| the machine at scale | 224 Gbit/s | 1000 logical qubits at distance 15, every µs: at $d^2-1 = 224$ syndrome bits per round per logical qubit, 1000 × 224 bits / 1 µs = 224 Gbit/s of syndrome bandwidth, plus a billion decode-instances/s that can never fall behind |

The result is more precise than the usual headline. It is not that decoding is impossible: for a *single* logical qubit, modern algorithmic decoders already beat the real-time budget (the bench measures tens of nanoseconds; the literature's FPGA decoder, 550 ns). The wall is that the machine must run those decoders for thousands of logical qubits every microsecond without ever falling behind, which is why the field builds pre-decoders[7] and distributed decoders. The bottleneck is not the qubits you can throw money at; it is a systems problem in classical, low-latency compute, and pure, readable Rust belongs there as much as anywhere. That is why this thread sits in a quantum topic on an *edge* lab's bench.

## 6 · How does the field divide into five families?

The threads above are ours to build; this one is already deployed, and it is worth stating plainly because it shows the quantum-inspired-on-classical-silicon pattern surviving contact with a fielded programme, on hardware of the class discussed here. Infleqtion's **SAPIENT** (Secure AI for PNT) won a **$2M U.S. Army award** after placing first among 133 entrants in the Army's xTechScalable AI competition. It fuses multiple sensor streams for assured navigation and timing when GPS is *denied or spoofed*, using quantum-inspired multimodal learning and **Boltzmann-machine** (that is, energy-based) models, deployed as small models on **NVIDIA Jetson edge GPUs**, with testing scheduled through October 2026.[9] Note carefully what is and is not quantum: the models' *heritage* is

quantum, the *deployment* is ordinary classical silicon riding on the vehicle. That is the shape of this entire field.

Surveyed across languages rather than English alone, the field resolves into five families, and the multilingual record carries news this review did not locate in the English-language sources it searched (arXiv, Nature and Science news, and the vendor press rooms cited below): in February 2026 Toshiba and Mirise reported *mounting a quantum-inspired optimizer (a simulated-bifurcation machine) onto an autonomous mobile robot*, billed as a world first for embedded real-time decision-making.[10]

| family | what it does for a body | maturity |
|---|---|---|
| tensor-network compression | fit a VLA / world model / policy on the robot's own chip | deployed & funded |
| Ising machines (SBM, annealer, photonic) | planning, scheduling, routing as energy minimization | real; now reaching robots |
| quantum-inspired planning metaheuristics | path and trajectory search, classically | emerging, edge-runnable |
| energy-based / Boltzmann fusion | multimodal sensing that holds under spoofing | deployed (Jetson) |
| quantum-inspired RL & control | compact policies, stability structure | early research |

We built the deployed pattern small enough to read, and measured it. The model is a conditional Boltzmann machine: a continuous state $x$ with one binary *trust* unit per sensor, whose energy charges the data mismatch of each trusted sensor and a flat $\lambda$ for distrusting one. Mean-field inference gives a Boltzmann trust weight $w_k = \sigma(\beta(\lambda - r_k))$ and fuses $x$ as the trust-weighted estimate. Over 40,000 trials with nine sensors:

| scenario | naive mean | median | energy-based | oracle* |
|---|---|---|---|---|
| 3 of 9 spoofed (12σ adversarial bias) | 2.36 | 0.85 | 0.60 | 0.58 |
| 4 of 9 merely degraded (5× noise) | 1.61 | 0.93 | 0.73 | 0.62 |
| 5 of 9 spoofed (corrupt majority) | 3.02 | 1.76 | 1.52 | 0.70 |

*The oracle is *told* which sensors are lying: not deployable, included as the best-achievable floor. The readings are these. Against hard spoofing the energy model nearly reaches the oracle and clearly beats the outlier-resistant median, because the trust units drive spoofed sensors' weight to zero. Against mere degradation it beats the median by a wider relative margin, since a median discards *how much* a sensor disagrees while soft weights use it. And in the third row we report our own failure: once the corrupted sensors are the **majority**, no fuser recovers, ours included. The model is nine trust units and two scalars, fusing in about 1.5 $\mu$s on a laptop-class core: far inside a control loop, on the hardware the robot already carries. You can run it live in your browser.

This thread has a consequence outside its own topic, and it is the sharpest thing we learned building it. A robot does not fuse sensors for their own sake; it fuses them so something downstream can decide whether an action is safe. Hand this same trust-weighted estimate to a **safety certificate** and the picture turns: a barrier evaluated on a captured state does not degrade gracefully, it becomes confidently wrong, clearing a body that is already inside the hazard. Worse, against a *coordinated* spoof the trust fuser above is not merely insufficient but actively harmful: it locks onto the self-consistent lie and produces *more* false clearances than a naive average. The repair is not a better estimator but a different output: detecting that a rival explanation exists and **refusing to certify**. That work is written up in *The Certificate Doctrine*, and the failure is watchable in the certificate under spoofing. It is a good example of why this lab keeps its quantum-inspired work next to its control work: the compression thread makes a model fit the robot, and this thread decides whether the robot should be believed.

## 7 · What bar does a quantum claim have to clear?

The reason this paper leads with quantum-*inspired* methods rather than quantum hardware is not timidity; it is the field's own most important result. Many quantum machine-learning speedups, once examined, can be matched by a classical algorithm given comparable data access (a process called **dequantization**[1]), and near-term practical advantage is confined to narrow, often contrived regimes. This is a gift, not a disappointment: it means the *structure* quantum methods reveal (low-

entanglement factorizations, Ising formulations, reservoir dynamics) is frequently the real source of the win, and that structure runs on classical hardware. The discipline of this topic is to find where the structure genuinely helps a body, to measure it, and to say plainly when it does not.

**Scope.** Every result here is a small bench in the lab's open record: one control policy and one weight matrix for the compression thread; a low-rank (two-core) factored kernel on-device, with the deeper quantized tensor-train chain measured separately (it wins on structured weights, not on random ones, and its matvec beats dense only at small bond); a small traveling-salesman QUBO with a generic classical solver. The compression win *requires* the layer to be genuinely low-rank, which (as the bench measures) means training the factored form, not squeezing a dense one; a random or well-sized dense layer barely compresses. The full on-device MAC saving assumes factored-format inference; that op is now fused into the lab's runtime as a first-class factored-linear layer, verified against the dense path and wall-clock-measured on the live GPU, but its measured speedup reaches the multiply-count ceiling only at batch width and is a smaller constant at single-token decode, where bandwidth and kernel-launch overhead dominate. The decoder thread (four) uses the repetition code as the canonical decoder proxy, validated against an exact analytic ground truth; the surface code scales the graph but not the shape of the argument, and the per-instance latency figures are for a single logical qubit; the systems wall is the aggregate across a full machine, which we report as arithmetic on verified device numbers, not as a decoder we ran at that scale. The fusion thread (five) is a nine-sensor 2-D benchmark with a synthetic spoofing model, not a field trial on a vehicle; it demonstrates the mechanism and its breaking point, and the deployed systems it mirrors are far larger. And no quantum hardware was used or is required anywhere in this paper: these are classical implementations of ideas that originated in quantum information science. The topic is in active, open exploration; this paper is its current state, not a closed case.

## What decides this

**Why lead with quantum-inspired methods rather than quantum hardware?** Because of the field's most useful result. Many quantum speedups can be matched classically once the same data access is allowed, a process called dequantization, and that finding is a gift. It says the *structure* these methods expose is frequently the real source of

the win: low-entanglement factorizations, Ising formulations, reservoir dynamics. Structure runs on the silicon a robot already carries.

**What is the bar?** An embodied workload where the quantum-inspired formulation beats the best *dequantized* classical algorithm, measured in joules per task on hardware a body can carry. Comparing against the dequantized version is what makes the result mean something for a machine in the world.

**What if a method is matched by its own dequantization?** Then the structure was the whole content, and it should be taken and run classically. That is a result to use rather than a disappointment.

**What opens up?** Planning and temporal computation on a robot at a fraction of today's energy, using formulations discovered by asking what a quantum machine would do and then noticing the answer runs on the machine already in hand.

## References & lineage


1. [1] E. Tang and collaborators, dequantization of quantum machine learning (Nature Reviews Physics, Dequantizing algorithms to understand quantum advantage in ML, 2022): the bar for near-term quantum advantage.
2. [2] Tensor networks for neural-network compression: Tensorization is a powerful but underexplored tool (2026); Tetra-AML (automatic tensor-network compression); hardware-aware tensor networks (real-time on FPGAs). The lab's own benches: `tt_policy_compression.py` (train-compact vs compress-after), `tt_kernel.rs` (verified factored matvec, MAC counts), and `qtt_chain.py` (the deep quantized tensor-train, measured on structured vs random weights); and `factored_linear.rs` (the factored op fused into the runtime, verified against the dense path and wall-clock-measured on the live GPU).
3. [3] Planning as QUBO/Ising: Quantum annealing for inverse kinematics (Nature Sci. Rep., 2026); Quantum annealing for combinatorial optimization (survey, 2026); A. Lucas, *Ising formulations of many NP problems* (2014). Bench: `qubo_planning.py`.
4. [4] Gate-based quantum reservoir computing on NISQ hardware (2026); edge of many-body chaos in quantum reservoir computing (2026); classical reservoir computing / echo-state networks are the edge-deployable shadow.
5. [5] Below-threshold quantum error correction: Google Quantum AI, Quantum error correction below the surface code threshold (Nature, 2024): the first below-threshold surface-code memory, with an integrated real-time decoder.
6. [6] Quantum-LDPC codes and real-time classical decoding: IBM's 2029 fault-tolerance roadmap (bivariate-bicycle qLDPC, ~10× overhead reduction, decoder running in real time on classical

hardware).

7. [7] The real-time decoder bottleneck: FPGA-based neural-network surface-code decoder (550 ns closed-loop inside a 1.25 µs cycle, 2026); a local pre-decoder to reduce bandwidth and latency; latency-constrained hardware-aware QEC co-design (2026). Bench: `qec_decoder_wall.rs` (LUT explosion, a decoder validated against exact analytics, latency, and the throughput wall).
8. [8] On reading the topological headlines: Microsoft's topological-qubit claim faces fresh challenge (Nature news, 2025): the 2025 paper carries an editorial caveat (reviewers found no evidence of Majorana zero modes); a separate 2021 Majorana paper was retracted. The two are often conflated.
9. [9] Deployed quantum-inspired / energy-based fusion at the edge: Infleqtion, $2M U.S. Army contract for Contextual Machine Learning in assured navigation and timing (SAPIENT; Boltzmann-machine-based multimodal fusion on NVIDIA Jetson), and CML at GTC 2025. Bench: `ebm_fusion.rs`; live sim: `/assets/sims/ebm-fusion`.
10. [10] The multilingual record: 東芝 / Mirise, 世界初、量子インスパイアード最適化計算機を自律移動ロボットに搭載 (a quantum-inspired optimizer mounted on an autonomous mobile robot, Feb 2026); NTT 100,000-spin coherent Ising machine; Multiverse Computing / CompactifAI (tensor-network model compression); RLRC (compressed VLA recovery); Lyapunov-aware quantum-inspired RL for continuous-time vehicle control.
11. Companion: the Quantum information at the edge topic, the live compression bench, and the qutrit emulator: beside the lab's open compute-commons work on ternary and thermodynamic computing.