Stored Episodic Evolutionary DNA Navigation (SEED-Nav): DNA-Encoded Memory, Genetic-Algorithm Adaptation, and LiDAR/SLAM-Ready Navigation

Darin R. Molnar, PhD
Shoreline Consulting Group
February 13, 2026

Abstract

Stored Episodic Evolutionary DNA Navigation (SEED-Nav) is an autonomous navigation architecture that treats a robot's experiential memory as virtual chromosomal deoxyribonucleic acid (DNA) strings and uses those memories to support robust, incremental behavior selection in uncertain, dynamic environments. The design emphasizes (a) an always-running reactive control loop for safety and obstacle clearance, (b) a modular "pluggable memory" subsystem where episodes/events/policies/maps are serialized into A/T/G/C chromosomes, (c) an evolutionary pathway for iteratively refining navigation policies using stored experience as selectable "genotypes," and (d) scan-based perception via low-cost 2D LiDAR that supports both immediate obstacle awareness and a pathway to LiDAR SLAM for mapping and localization. In its implemented minimum viable product (MVP), that pathway is realized as a lightweight on-device genetic algorithm (GA) that evolves bounded navigation parameters across repeated episodes using selection, recombination, and mutation, with fitness computed from DNA-encoded episode summaries. In addition, post-hoc analysis of chromosome logs produces environment "map deltas" (e.g., hazard/hotspot grids) stored as chromosomes and reused to bias subsequent exploration, while preserving offline operation during roving episodes. This paper summarizes the salient architectural choices, describes a practical DNA encoding scheme for memory persistence, and situates the approach relative to behavior-based robotics, probabilistic robotics/SLAM, and evolutionary robotics.

Introduction

Navigational autonomy remains challenging because sensing is partial, geometry is cluttered, and dynamics are unpredictable. Classical robotics often frames navigation as estimation and control under uncertainty, typically requiring careful modeling of sensors, motion, and environment. Probabilistic robotics formalizes this with Bayesian state estimation and decision-making, and remains foundational for LiDAR-based SLAM and localization-driven navigation pipelines. [3], [8]

In parallel, behavior-based robotics demonstrates that robust autonomy can emerge from layered reactive control where sensing is tightly coupled to action, often with minimal world modeling. The subsumption tradition highlights practical resilience: agents can exhibit useful autonomy early, while additional layers add competence without breaking baseline safety behaviors. [1], [2]

SEED-Nav proposes a hybrid: reactive control for safety and obstacle clearance plus a memory substrate that records navigation episodes, policy deltas, and (optionally) map deltas as virtual deoxyribonucleic acid (DNA) chromosomes. With a low-cost 2D LiDAR added to the perception stack, SEED-Nav can support scan-based obstacle awareness immediately and provides a straightforward upgrade path to SLAM-based localization for improved homing and

mapping. Over time, the system's "genetic" policy parameters—and post-hoc environment representations derived from episode corpora—can be refined using selection criteria grounded in real runs, and those updates can themselves be stored as selectable chromosomes. This aligns conceptually with evolutionary robotics and genetic search over controller parameters. [4], [6], [7], [8]

System overview

At a high level, SEED-Nav comprises six interacting layers:

1. **Perception layer (touchless proximity + LiDAR + telemetry)**
   For indoor navigation minimum viable products (MVPs), touchless ranging provides high-frequency safety signals while avoiding mechanical bump sensors. In the current build, three ToF sensors are purposefully oriented as two downward-facing "cliff" detectors (stairs/ledges) and one upward-facing detector for overhangs and upper-edge obstacles. A low-cost 2D LiDAR provides dense range scans (e.g., forward-arc minima) for obstacle triggering and supports scan-based mapping/localization when compute permits. Battery telemetry supports safety triggers and "return-to-home" behavior (RTH).

2. **LiDAR/SLAM layer (scan processing + mapping/localization)**
   LiDAR scans can be reduced to safety-critical features (e.g., minimum range in a forward arc) for deterministic avoidance, and—when desired—fed into a 2D SLAM pipeline (scan matching, occupancy-grid mapping, loop closure) to improve localization and enable map-based homing. [3], [8], [9]

3. **Reactive safety and clearance layer (always on)**
   A high-priority controller implements immediate obstacle handling (brake → back → turn → probe), ensuring the robot attempts to clear obstacles rather than only stopping. This is directly in the spirit of layered reactive control: collision avoidance remains operational regardless of higher-level planning. [1]

4. **Navigation and task layer (explore, return-home, stop)**
   A mode manager coordinates exploration, RTH on low-battery conditions, and hard stops for safety. In an MVP, RTH can be implemented using dead-reckoned pose (integrating commanded motion); with LiDAR enabled, later phases can replace or correct drift using scan-based localization/SLAM and map-based homing. [3], [8]

5. **Chromosomal memory layer (DNA strings + metadata index)**
   Every salient event (startup, obstacle encountered, RTH trigger, arrival, etc.) is committed as a chromosome: a canonical A/C/G/T string encoding the memory payload, plus a lightweight metadata index for retrieval. This mirrors DNA data storage concepts—mapping digital information to nucleotide alphabets—while adapting it to embedded robotics constraints (local persistence, fast append, simple verification). [5]

6. **Evolutionary adaptation layer (GA over parameters + post-hoc space modeling)** Navigation parameter sets are treated as genomes and evaluated over bounded episodes. Each episode produces DNA-encoded summaries used to compute fitness; the GA applies elitism, recombination (crossover), and bounded mutation to produce the next generation. Post-hoc aggregation of episode events can also produce MAP_DELTA "environment genes" (e.g., hazard/hotspot grids) that are stored as chromosomes and reused to bias future exploration. [6], [7]

Methods: Mathematical and Algorithmic Specification

M1 Layered autonomy architecture

Foundations and antecedents: subsumption/behavior-based control [1], [2]

SEED-Nav is formalized as a layered autonomy stack $\mathcal{A} = \langle L0,\ldots,L5\rangle$: L0 safety/reflex control, L1 reactive navigation primitives, L2 perception (ToF + 2D LiDAR scan processing), L3 mapping/localization (optional SLAM), L4 episodic "DNA" memory, and L5 evolutionary adaptation (GA over parameters plus post-hoc environment deltas). Strict priority arbitration ensures that safety remains operational even if higher layers fail.

M2 State, dynamics, and sensor models

Probabilistic state estimation background [3]

Let the robot pose be $x_t = [x_t, y_t, \theta_t]^T \in SE(2)$ with control $u_t = [v_t, \omega_t]^T$ (linear and angular velocity). A discrete-time kinematic model is $x_{t+1} = f(x_t, u_t) + w_t$ (1) where w_t captures process noise. In an MVP without wheel encoders, dead-reckoning integrates commanded velocities to attach approximate poses to events and to support bounded return-to-home behavior.

Near-field Time-of-Flight (ToF) sensors provide safety-critical ranges. Two downward sensors $r\downarrow_{t,L}$, $r\downarrow_{t,R}$ detect cliffs/stairs; one upward sensor $r\uparrow_t$ detects overhead hazards.

A conservative cliff gate is

$$\text{cliff} \Leftrightarrow \max(r\downarrow_{t,L}, r\downarrow_{t,R}) \geq r_{floor} + \Delta_{cliff} \quad (2)$$

A 2D LiDAR scan is

$$Z_t = \{(\varphi_i, \rho_i, I_i)\}_{i=1}^{N}.$$

For immediate avoidance, a forward-arc minimum range is computed as

$$\rho^{front}_t = \min_{\varphi_i \in [-\alpha, \alpha]} \rho_i \quad (3)$$

and avoidance triggers when $\rho^{front}_t \leq \rho_{obs}$.

M3 Reactive safety and obstacle clearance

SEED-Nav implements a deterministic, always-on clearance routine that can be modeled as a timed hybrid automaton with phases {BRAKE, BACK, TURN, PROBE}. Each phase applies constant control for a bounded duration τ, producing repeatable behavior while still allowing parameters (e.g., τ and speed caps) to be optimized by the GA.

M4 Mapping and SLAM pathway

SLAM and mapping background (tutorial + occupancy grids + particle-filter SLAM + scan-matching loop closure) [8], [11], [12], [10], [9]

SEED-Nav treats LiDAR as both an immediate scan-based obstacle sensor and a modular entry point for SLAM. When mapping is enabled, an occupancy grid map M is estimated using log-odds updates.

$$\ell_{i,t} = \ell_{i,t-1} + \log(P(m_i=1 \mid x_t, Z_t) / (1 - P(m_i=1 \mid x_t, Z_t))) - \ell_0 \quad (4)$$

Full SLAM targets the posterior $P(x_{1:t}, M \mid u_{1:t}, Z_{1:t})$ and may be realized with RBPF grid-based SLAM or pose-graph optimization with scan matching and loop closure. The architectural requirement is plug-in compatibility: enabling SLAM must not compromise L0 –L2 safety behaviors.

M5 Episodic memory as DNA chromosomes

DNA as an information medium and robust encoding for storage [5]

SEED-Nav uses $\Sigma=\{A,T,C,G\}$ as a base-4 alphabet for canonical, append-only persistence. Bytes are encoded by mapping each 2-bit chunk to a nucleotide. A chromosome stores an envelope (schema/version, length, checksum) and a payload (e.g., JSON/CBOR, optionally compressed). Complementary base-pairing (A–T, C–G) is not required in this digital representation, but may be used for redundancy or error-correction if desired.

Memory items are typed records (EVENT, EPISODE, POLICY_NOTE, MAP_DELTA, …) with a metadata index for retrieval. Chromosomes remain the canonical artifact; indices enable fast filtering while deferring decoding.

M6 Genetic algorithm for policy-parameter evolution

Genetic algorithms and evolutionary robotics foundations [6], [7], [4]

Define a bounded parameter vector $\theta \in \mathbb{R}^d$ that encodes navigation thresholds and timed-automaton durations (e.g., ρ_{obs}, ρ_{clear}, τ_{back}, τ_{turn}, v_{max}). Each episode produces summary statistics (progress proxy, obstacle/cliff events, avoidance time, return-to-home success) persisted as EPISODE chromosomes.

A shaped scalar fitness for episode e is

$$F(e) = w_L L_e - w_O N^{obs}_e - w_C N^{cliff}_e - w_A T^{avoid}_e + w_H I^{home}_e \quad (5)$$

Selection (e.g., tournament), recombination (uniform crossover), mutation (bounded Gaussian perturbations), and elitism define the evolutionary update. Safety is enforced by hard bounds and by L0 gates, preventing evolved parameters from violating collision/cliff constraints.

M7 Post-hoc evolution of the navigation space

SEED-Nav extends evolution beyond parameters by deriving reusable environment structure from logs. Let H be a hazard/hotspot grid accumulated from obstacle/cliff events:

$$H_{i,j} = \Sigma_e \Sigma_{t \in e} \, 1[(x_t,y_t) \in cell(i,j)] \cdot 1[event_t \in \{obstacle, cliff\}] \quad (6)$$

These "map deltas" are stored as MAP_DELTA chromosomes and reused to bias subsequent exploration away from high-penalty cells, improving efficiency while preserving offline-first execution during roving.

M8 Optional AI in post-navigation analysis

Because the full episode history is persisted, heavier analyses can be run after missions on-device (when docked), on a workstation, or via an AI API when connectivity is available. Examples include clustering recurrent obstacle motifs, generating natural-language mission summaries, detecting drivetrain anomalies from telemetry, or re-running SLAM offline with higher compute budgets. Mission execution remains deterministic and local.

M9 Evaluation protocol

A practical evaluation uses repeated fixed-duration household episodes. Report (a) learning curves of mean/best fitness across GA generations, (b) ablations isolating crossover and mutation effects, (c) return-to-home success under battery triggers, and (d) reductions in repeated obstacle encounters after MAP_DELTA reuse. When SLAM is enabled, measure map consistency and localization drift relative to available ground truth.

DNA-encoded memory representation

Rationale

SEED-Nav's "chromosomal memory" is primarily an architectural metaphor with practical benefits:

- **Uniform persistence format:** anything storable becomes a chromosome (episodes, maps, policies, skills).

- **Transportable and inspectable:** chromosomes are text strings (A/T/G/C) that can be diffed, logged, replicated, and archived.
- **Error detection:** embedded systems benefit from explicit integrity checks.
- **Evolution metaphor alignment:** "genotype" representations (chromosomes) can be selected, mutated, recombined, and replayed.

This is consistent with broader DNA storage research, which demonstrates robust encoding/decoding of digital artifacts into nucleotide sequences and emphasizes error management and efficient packing. [5]

Practical encoding (MVP)

A practical embedded-friendly scheme is:

- **Payload:** JSON (or another compact serialization) for a memory item: kind (EPISODE/EVENT/POLICY_NOTE/MAP_DELTA/SKILL), timestamp, tags, and body.
- **Compression:** optional zlib compression to reduce chromosome length.
- **Envelope:** header (magic/version/schema/flags), payload length, CRC32 checksum, followed by payload bytes.
- **Base-4 mapping:** encode each byte as 4 nucleotides using a fixed 2-bit → {A,T,G,C} mapping.

This yields deterministic chromosomes that can be verified using cyclical redundancy check (CRC) methods and decoded locally without external tooling. While DNA storage research often focuses on biochemical constraints (homopolymers, GC content, sequencing noise), the SEED-Nav MVP uses "DNA" as a semantic and computational encoding, not as a wet-lab substrate. DNA Fountain and related work motivate the general feasibility and robustness concepts when mapping digital information to nucleotide alphabets. [5]

Retrieval strategy

A two-tier retrieval strategy supports efficiency:

- **Metadata-first:** store a JSONL index with id, kind, timestamp, and tags for fast filtering.
- **Decode on demand:** only decode chromosomes required by the query.

Later phases can incorporate similarity search over DNA strings (e.g., k-mer/minimizer-style hashing) or hybrid approaches with embedding "genes," but metadata-first retrieval is sufficient for an MVP and aligns with embedded constraints.

Control loop design

SEED-Nav's runtime loop is intentionally structured to privilege safety and deterministic behavior:

1. **Sense**: read ToF distances (downward cliff sensors + upward overhead sensor), LiDAR scan features (e.g., forward-arc minimum range), battery voltage/current, and time.
2. **Update state**: update pose estimate (dead-reckoning in MVP; optionally corrected by LiDAR scan matching/SLAM when enabled). [3], [8]
3. **Safety gates**: enforce hard-stop thresholds, motor failsafes, and cliff-prevention triggers (downward ToF) as the highest-priority safety behavior.
4. **LiDAR preprocessing**: compute obstacle triggers from scan arcs/segments so reactive clearance remains deterministic even when SLAM is disabled.
5. **Priority arbitration:** If a cliff trigger fires (downward ToF indicates a drop), execute the cliff routine (brake → back → turn) until safe.
    - If an obstacle trigger fires, run the clearance routine until resolved (or timeout).
    - Else, if battery condition crosses threshold, switch to RETURN_HOME mode.
    - Else, remain in EXPLORE mode.
6. **Actuate:** apply motor commands with slew-rate limiting (reduces jerk and noise for "quiet/slow" motion).
7. **Commit memory:** log salient events as chromosomes.
8. **Episode scoring (optional)**: compute fitness from logged events, update GA state for the next run, and optionally generate post-hoc MAP_DELTA chromosomes (e.g., hazard grids) from the episode corpus. [6], [7]

This structure is closely aligned with behavior-based robotics principles: immediate sensor-action coupling for safety and robustness, with higher-level modes layered above it. [1]

Evolutionary metaphor and learning pathway

Evolutionary robotics provides the conceptual bridge between "chromosome memory" and adaptive autonomy: controllers and parameters can be represented as genomes and refined via selection on real or simulated episodes. [4]

In SEED-Nav terms:

- **Genotype:** a chromosome (or set of chromosomes) encoding navigation parameters (speed caps, avoidance thresholds, turn durations), plus optional policy rules.
- **Phenotype:** observed behavior in the home environment (coverage, collisions avoided, time-to-return, energy efficiency).
- **Fitness:** a scalar or multi-objective score derived from episode logs (chromosomes).
- **Selection/variation**: choose top-performing parameter sets, apply recombination (crossover) and bounded mutation, and redeploy. [6], [7]

Implemented adaptation: GA over parameters

In the current SEED-Nav MVP, the evolutionary metaphor is made concrete via a simple genetic algorithm that searches over a bounded vector of numeric navigation parameters (e.g., obstacle and clearance thresholds, back/turn/probe timings, speed caps, and slew limits) [6], [7]. Each parameter vector is a candidate genome.

The robot runs an exploration episode (episodic evaluation) for a fixed time budget (or until a safety/battery condition triggers RETURN_HOME), then returns to its start pose using the same deterministic controller backbone. An EPISODE chromosome records key metrics (e.g., obstacle encounter rate, avoidance time, distance proxy, cliff triggers, and return success), and fitness is computed from those metrics. The GA retains an elite subset and generates new candidates via recombination (crossover) plus bounded mutation; clamping and safety gates ensure evolved parameters cannot exceed conservative navigational limits.

The architecture does not require cloud AI to function; it can evolve purely from onboard experience. External AI services may still be useful for *semantic labeling* (e.g., converting logs to natural-language summaries or extracting "room types"), but autonomy is retained by keeping the control loop and memory local.

Mitigating premature convergence

Because small populations and noisy episode-based fitness can cause early collapse to a suboptimal parameter basin, SEED-Nav explicitly preserves genetic diversity while retaining selection pressure. Parent selection is performed via tournament selection from a widened parent pool (rather than only the elites), each generation injects a small fraction of random "immigrant" genomes, and mutation strength is adaptively increased when population diversity falls below a threshold (diversity-triggered mutation boost). Elitism is retained for stability, but diversity preservation mechanisms ensure continued exploration of the bounded parameter space and reduce sensitivity to lucky early episodes. These strategies are standard diversity-maintenance measures in evolutionary computation and are particularly important for embodied evaluation where fitness estimates are stochastic and expensive. [7]

Post-hoc evolution of the navigation space (environment "genes")

Beyond parameter tuning, SEED-Nav can evolve an explicit representation of the navigation space in a post-hoc manner. After a batch of episodes (e.g., a GA generation), DNA-encoded obstacle and cliff events are aggregated into a coarse hazard/hotspot grid. The resulting MAP_DELTA chromosome becomes an inspectable "environment gene" that can be selected and reused to bias subsequent exploration (e.g., turning away from repeatedly problematic cells), providing incremental environmental knowledge without requiring continuous SLAM during the real-time safety loop. If additional compute is available after episodes, LiDAR-based SLAM or external AI services can be applied to the stored chromosome corpus for richer map building, semantic labeling, or failure-mode clustering—without requiring Wi-Fi connectivity during roving.

Strengths, limitations, and future work

Strengths

SEED-Nav's most distinctive contribution is the explicit coupling of (a) layered reactive control, (b) a canonical chromosome memory substrate, (c) a GA that refines bounded navigation parameters via recombination and mutation, and (d) LiDAR/SLAM-ready perception that can improve obstacle handling and enable map-based homing. The architecture is implementable on modest compute, is inspectable (text chromosomes), and supports incremental refinement while keeping real-time safety behaviors always on. [6], [7], [8]

Limitations (MVP)

The current MVP can operate with dead-reckoning for RTH and use LiDAR primarily for obstacle triggering; both choices can be sufficient for short indoor runs but will drift over time and do not provide globally consistent maps. Robust homing and repeatable room-to-room navigation typically benefit from scan matching and SLAM-style localization with loop closure and map maintenance. [3], [8], [9]

Future work

High-leverage extensions include: (a) integrating full 2D LiDAR SLAM (scan matching + loop closure) for map-based homing and coverage planning, (b) richer chromosome schemas for maps and policies (TLV "gene" segments, ECC), (c) multi-objective GA operators and richer fitness metrics grounded in coverage and energy efficiency, (d) formal safety envelopes so evolutionary updates cannot violate collision, stall, or speed constraints, and (e) post-hoc application of AI to DNA-encoded episode logs to generate human-readable summaries, cluster recurrent failure motifs, and propose testable policy/map deltas—while keeping the real-time loop deterministic and offline during episodes. [8], [9], [10]

Conclusion

SEED-Nav is a practical, modular architecture for autonomous navigation that treats memory as virtual DNA chromosomes and uses layered reactive control as the safety backbone. It is positioned to evolve navigation behavior across repeated navigation episodes by making experience persistently selectable. By combining behavior-based robustness with an explicit genomic memory representation inspired by DNA data storage and evolutionary robotics, SEED-Nav offers a coherent foundation for incremental autonomy on low-cost hardware.

References

[1] I. Ögren and N. Sprague, “Behavior Trees in Robot Control Systems,” Annual Review of Control, Robotics, and Autonomous Systems, vol. 5, pp. 81–107, 2022, doi:10.1146/annurev-control-042920-095314.

[2] A. Waga, S. Benhlima, A. Bekri, J. Abdouni, and F. Z. Saber, “A survey on autonomous navigation for mobile robots: From traditional techniques to deep learning and large language models,” Journal of King Saud University – Computer and Information Sciences, vol. 37, art. 198, 2025, doi:10.1007/s44443-025-00216-x.

[3] M. Abdelkarim, H. Voos, and D. Gorges, “A tutorial and review on simultaneous localization and mapping based on factor graph optimization,” IEEE Access, vol. 13, pp. 204168–204219, 2025, doi:10.1109/ACCESS.2025.3540627.

[4] J. T. Carvalho and S. Nolfi, “The Role of Morphological Variation in Evolutionary Robotics: Maximizing Performance and Robustness,” Evolutionary Computation, vol. 32, no. 2, pp. 125–142, 2024, doi:10.1162/evco_a_00336.

[5] X. Zhang and F. Zhou, “An Encoding Table Corresponding to ASCII Codes for DNA Data Storage and a New Error Correction Method HMSA,” IEEE Transactions on Nanobioscience, vol. 23, no. 2, pp. 344–354, 2024, doi:10.1109/TNB.2024.3356522.

[6] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Ann Arbor, MI, USA: University of Michigan Press, 1975.

[7] V. A. Cicirello, “Evolutionary Computation: Theories, Techniques, and Applications,” Applied Sciences, vol. 14, no. 6, art. 2542, 2024, doi:10.3390/app14062542.

[8] Y. Ran, X. Xu, Z. Tan, and M. Luo, “A Review of 2D Lidar SLAM Research,” Remote Sensing, vol. 17, no. 7, art. 1214, 2025, doi:10.3390/rs17071214.

[9] S. Macenski and T. Jambrecic, “SLAM Toolbox: SLAM for the dynamic world,” Journal of Open Source Software, vol. 6, no. 61, art. 2783, 2021, doi:10.21105/joss.02783.

[10] İ. İnce, D. Yiltas-Kaplan, and F. Keleş, “From Simulation to Reality: Comparative Performance Analysis of SLAM Toolbox and Cartographer in ROS 2,” Electronics, vol. 14, no. 24, art. 4822, 2025, doi:10.3390/electronics14244822.

[11] P. Racinskis, J. Arents, and M. Greitans, “Constructing Maps for Autonomous Robotics: An Introductory Conceptual Overview,” Electronics, vol. 12, no. 13, art. 2925, 2023, doi:10.3390/electronics12132925.

[12] Y. Liu, W. Zhang, F. Li, Z. Zuo, and Q. Huang, “Real-Time Lidar Odometry and Mapping with Loop Closure,” Sensors, vol. 22, no. 12, art. 4373, 2022, doi:10.3390/s22124373.