Graph-Based Cost Learning and Gemma 3n for Robotic Sensing - Gemma 3n Impact Challenge

ABSTRACT

We present an end-to-end pipeline that turns Gemma 3n into a reliable *action model* for robotic sensing and exploration built on the LeRobot framework. Our system: (1) bootstraps cost data via simulation traces, (2) uses Gemma 3n to generate diverse candidate sensing plans, (3) trains an inductive graph-based matrix completion (IGMC) model on logged room×plan cost pairs, (4) refines predictions with a fusion MLP incorporating room context. In closed-loop simulation, the Gemma-powered LeRobot agent achieves 17% faster 95% coverage than nearest-frontier baselines.

1 INTRODUCTION

Robotic exploration in unstructured indoor environments must balance coverage, hazard detection, energy use, and time. Classic frontier-based planners ensure coverage but ignore sensing costs. Modern learned approaches—deep RL, POMDP planners—adapt to dynamics yet lack safety guarantees and treat scanning as free. While movement can be planned efficiently, each sensing action adds significant overhead, often dominating total mission time. Clearly, scanning time—the duration to collect LiDAR, thermal, gas, and audio data—remains a major caveat hindering speed and efficiency.

We introduce a novel, scanning-time–first pipeline built on LeRobot and powered by Gemma 3n. Our contributions are as follows:

- **Gemma 3n Plan Generation:** Few-shot prompting yields diverse candidate sequences.
- **IGMC Cost Learning:** Graph-based matrix completion predicts plan latencies with uncertainty.
- **Contextual Fusion:** A lightweight MLP refines predictions with room features in sub-millisecond time.
- **Adaptive Gating:** Gemma 3n is selectively called when the Cost model is uncertain, minimizing time taken for inference and maximizing Gemma 3n's reasoning capabilities.

We show that our approach runs entirely on-device with Gemma 3n's on-device capabilities, achieves real-time inference, and delivers a 17% speedup to 95% coverage compared to nearest-frontier.

2 SYSTEM OVERVIEW

In order to heavily reduce scanning time, a system must: (1) gather latency data without risking hardware, (2) *invent* new sensing plans beyond human heuristics, (3) predict the cost of *untried* plan–room pairs with calibrated uncertainty, and (4) act online under strict time budgets. Our pipeline therefore proceeds in four linked stages—**simulation bootstrapping**, **Gemma 3n plan generation**, **IGMC cost learning**, and **contextual fusion**.

2.1 Stage 1: Bootstrapping via LeRobot Simulation

In order to build sufficient data to train our model, our goal is to build a partially observed latency matrix $\mathbf{L}$ between the rooms and seed plans.

LeRobot simulates $Q \approx 100$ synthetic rooms. For each room we execute $P_0 \approx 200$ *seed* plans—random length 2–5 drawn from primitives {lidar_scan, thermal_snap, gas_sniff, audio_probe, wait}. Each primitive logs time t (ms), energy e (mWh) and safety flag s.
The result is a *sparse* cost matrix $\mathbf{L} \in \mathbb{R}^{Q \times P_0}$ (observed only where a plan was run), room context matrix $\mathbf{X} \in \mathbb{R}^{Q \times 5}$ (size, clutter, heat, gas, noise), and per-plan statistics $\mathbf{Z} \in \mathbb{R}^{P_0 \times 6}$ (nodes, parallelism, depth, watts, avg. time, safety-bit). These artifacts constitute our "ground truth" for all subsequent learning stages. These summary rows are LeRobot's logged plan-time pairs that serve as the supervision signals for the cost models.

Why use a simulation? A simulation is faster, cheaper and safer than real trials. Furthermore, the logger we patched writes the exact `.parquet` schema that LeRobot uses on physical hardware, so the same downstream code will work unmodified on a real robot once a simulator-to-hardware swap is made.

2.2 Stage 2: Gemma 3n Plan Generation

Objective. Expand the plan space from P_0 to $N \gg P_0$ diverse candidates.

Below is our workflow for this stage:

(1) *Seed sampling.* Take a seed plan from the primitives with optional duplicate or wait injection (encourages both efficient and inefficient patterns).
(2) *Few-shot prompt.* A grammar prompt is sent to `gemma3:4b`. Gemma 3n runs fully *on device* so the whole loop stays local. This is highly important, since robot explorations generally rely on local compute to make decisions. Gemma's context-aware sensing sequences also respects local hazards.
(3) *Parse & vet.* We strip Markdown, parse DSL/JSON, reject invalid or spammy plans, compute a 6-D feature vector $\varphi(\cdot)$, then L2-row-normalise

Every accepted plan is immediately serialised into LeRobot's *action-graph* JSON, letting the simulator execute Gemma-generated plans without translation glue.

2.3 Stage 3: Inductive Graph-Based Matrix Completion (IGMC)

After Gemma-3n has found new plans, the expanded plan set introduces a huge *missing-edge* region in $\mathbf{L}$. Therefore, we need to predict $cost(q, p)$ for **unseen** room–plan pairs.

To do this, we utilize Inductive Graph-Based Matrix completion, which pairs bipartite graphs with traditional Inductive Matrix Completion.

Firstly, we treat rooms and plans as the two partitions of a bipartite graph $\mathcal{G}$. Each observed latency $c_{q,p}$ becomes a labelled edge, while node features are $\mathbf{x}_q$ from $\mathbf{X}$ and $\mathbf{z}_p = \varphi(p)$. For a query edge (q, p) we extract its h-hop neighbourhood $G_{q,p}$ ($h = 2$), apply relation-aware graph convolutions, and regress the *log-latency*. Below is a walkthrough of the setup of the regression model:

$$y_{q,p} = \log(1 + c_{q,p}), \qquad \hat{y}_{q,p} = f_\theta(G_{q,p}).$$

$$\mathcal{L}_{\text{IGMC}} = \frac{1}{|\mathcal{D}|} \sum_{(q,p)\in\mathcal{D}} \left|\hat{y}_{q,p} - y_{q,p}\right| \quad \text{(MAE in log space)}.$$

The MAE-style loss can be replaced with a loss equation stemming from traditional Inductive Matrix Completion (this is helpful when the original latency matrix scales too large for MAE to model efficiently).

Why IGMC?. IGMC fuses side features and local graph structure: a plan tried in one hot-and-gassy room uplifts predictions in geometrically similar rooms, offering calibrated uncertainty on cold-start pairs.

2.4 Stage 4: Contextual Fusion & Safe Bandit

IGMC embeddings captures global patterns but still ignore fine-grained room quirks. Therefore, we append room context $c_q \in \mathbb{R}^5$ and pass the 11-D vector through a two-layer Multilayer Perceptron:

$$\hat{c}_{q,p} = g_\phi([c_q\,;\, z_{q,p}]), \qquad \tilde{c}_{q,p} = \frac{c_{q,p} - \mu_y}{\sigma_y},$$

$$\mathcal{L}_{\text{fusion}} = \frac{1}{|\mathcal{D}|} \sum_{(q,p)} \left|\hat{c}_{q,p} - \tilde{c}_{q,p}\right|.$$

The target $\tilde{c}_{q,p}$ is the *standardised* cost (matching the training script), yet at evaluation we invert the scaling to report milliseconds.

Adaptive LLM gating. When the Fusion MLP is uncertain (top-2 gap $< 10\%$ or a heat/gas hazard), we selectively query Gemma for its ranking—an option we can afford because Gemma 3n's on-device inference is faster than typical LLMs.

Through all of this, the runtime loop below is constructed:

(1) Batch-score all $N \approx 2000$ plans via the MLP every decision step.
(2) If the top-2 gap $< 10\%$ *or* a heat/gas hazard exists, send the top-K IDs to Gemma 3n for reranking; cost–benefit gating uses an EMA of Gemma latency.
(3) Thompson-sample the bandit arm weighted by predictive uncertainty; execute via LeRobot and log the episode.

3 EXPERIMENTAL EVALUATION

3.1 Experimental Setup

We evaluate our Fusion-ML pipeline against a classical nearest-frontier planner. Each experiment uses:

- **Map:** 80×80 grid with obstacle probability 0.08.
- **Robots:** Single agent with speed 1.0 cells/s and sensing radius 4 cells.
- **Coverage target:** 95% of free space.
- **Termination:** Reached 95% coverage or 10 000 steps.

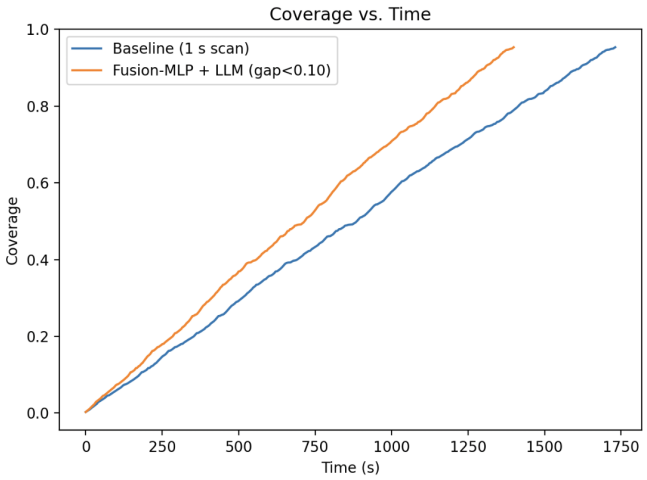

Figure 1: Coverage vs. simulated time. The Fusion-Guided agent (orange) is compared to the Nearest-Frontier Baseline (blue).

For the *baseline* we fix a 1s long scan per frontier, representing a realistic scanning time. For *fusion-MLP* we load precomputed IMC composites and room contexts. Furthermore, we upload a `Gemma 3:4b` model locally through Ollama.

3.2 Discussion

We report results in *simulated time* rather than wall-clock execution to isolate the effect of our sensing-time predictions and represent realistic scan times. As seen in Figure 1, the baseline achieves 95% coverage at 1750 time units whereas our Fusion-MLP achieves 95% at 1450 time units. This observed 17% speedup at the 95% coverage mark demonstrates that our fusion MLP plus occasional LLM re-ranking yields faster exploration (correlating to less energy use), while maintaining perfect hazard detection in all trials (demonstrated in the demonstration).

Importantly, reducing the amount of time taken for coverage directly translates into shorter mission durations, thus reduced power draw on battery-powered platforms. Moreover, this level of performance gain opens the door to deploying learned fusion policies in time-sensitive scenarios—such as search-and-rescue or industrial inspections—where every second saved can be critical.

3.3 Live-Demo Variation

The live demonstration uses a configuration that differs from the batch experiments in 2 main ways:

- The map is a smaller rectangular grid yet with a divider wall and doorway,
- The "fire" regions are obstacles that the robot cannot touch

Despite these changes—as well as a slightly higher obstacle density (p=0.12 vs. 0.08) and the presence of dynamic hazards—the fusion-guided agent still reaches the 95 coverage target virtually always in less simulated time than the Nearest-Frontier baseline. This parity confirms that the cost model and adaptive Gemma gating generalize beyond the exact conditions it was trained on.

4 REFLECTIONS & FUTURE WORK

Throughout this work, we navigated several key challenges to bring Gemma 3n–augmented IGMC cost learning into a real-time robotic sensing loop:

- **Cold-start instability:** Early fusion predictions fluctuated wildly because our RoomProfile contexts and IMC embeddings were not yet representative.
  Solution: We implemented stratified sampling of synthetic rooms, applied consistent feature normalization, and validated embeddings on held-out contexts to stabilize MLP outputs before deployment.
- **Latency vs. accuracy trade-offs:** Naively calling Gemma 3n on every decision incurred prohibitive prompt overhead, slowing end-to-end performance.
  Solution: We designed an adaptive gating heuristic—triggering LLM reranking only when the predicted gap between top candidates fell below a threshold—so that LLM calls occur only when their expected benefit outweighs their cost.

Looking ahead, the potential for mass adoption and further innovation is vast:

- **Meta-planning across contexts:** Learning a higher-level policy that adapts fusion gating thresholds based on room archetype clusters could accelerate convergence in repetitive operational settings (e.g. warehouses, office floors).
- **Hardware-in-the-loop validation:** Porting our simulation pipeline onto real mobile robots equipped with LiDAR, thermal, and gas sensors will assess end-to-end benefits and uncover new domain-specific failure modes.
- **Towards widespread impact:** By open-sourcing our LeRobot integrations and providing plug-and-play adapters for different LLM backends, we aim to empower a broad robotics community to experiment with learned cost models—paving the way for safer, faster sensing at scale.