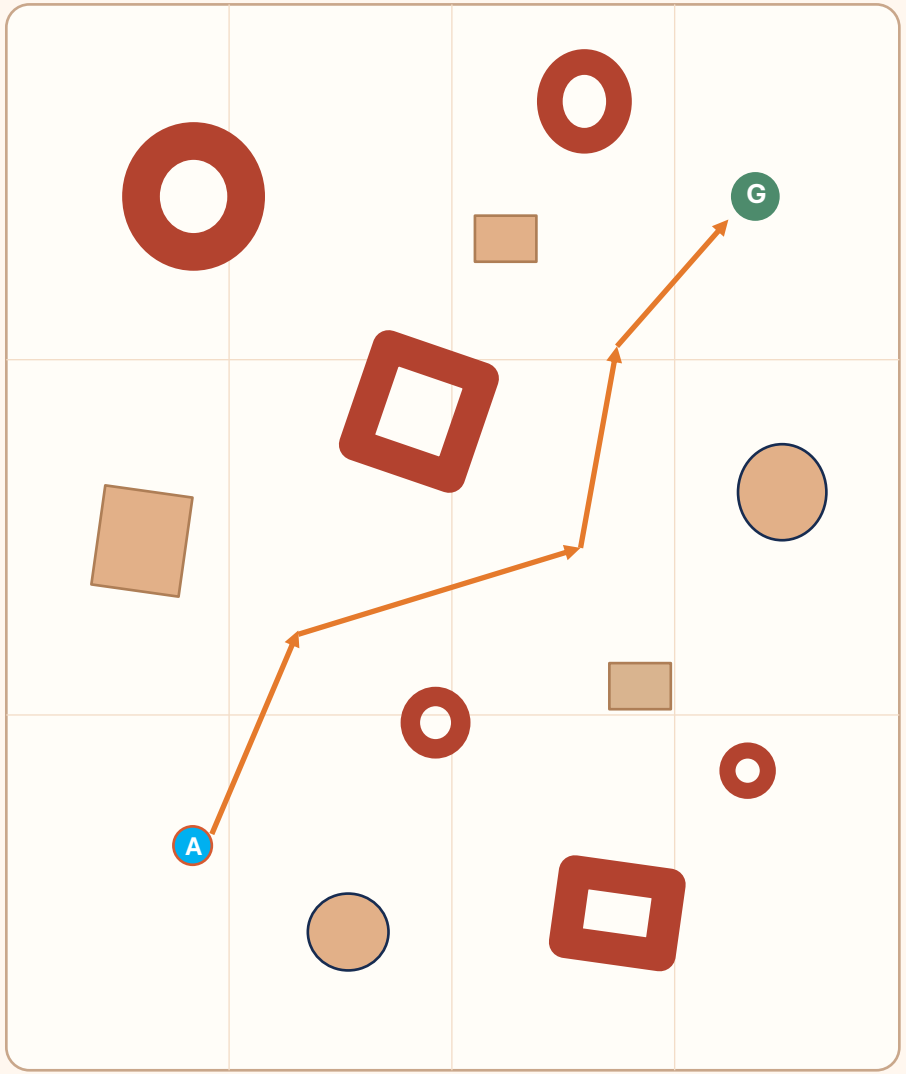
RL-Nav v1

Oracle-guided reinforcement learning for navigation under partial observability

Thesis: a blind agent can follow dense oracle embeddings and learn meaningful navigation behavior by following oracle commands.

2	512-D	62%
training stages	oracle embedding	policy success

Lorenzo Marinelli – 2043092
Reinforcement Learning 25/26



Random map with obstacles, pits, noisy motion, and hidden goal

MOTIVATION

Why this problem is interesting

The offloading of planning to the LLM

What the agent does not know

- It cannot directly observe the goal location.
- Local sensing is only five binary hazard sectors.
- Advice is refreshed every 16 steps, not every action.
- Slippage and collisions make execution inconsistent.

"Can a policy learn to execute abstract guidance consistently over time, even when it cannot directly see the goal?"

Partial observability

No direct goal view for the agent.
Only local cues

Omniscent Oracle

LLM has complete view of world map and agent state

Stochastic control

Linear + lateral slip
Angular noise, agent must learn to compensate

Temporal consistency

Advice ages
Memory matters

Research contribution

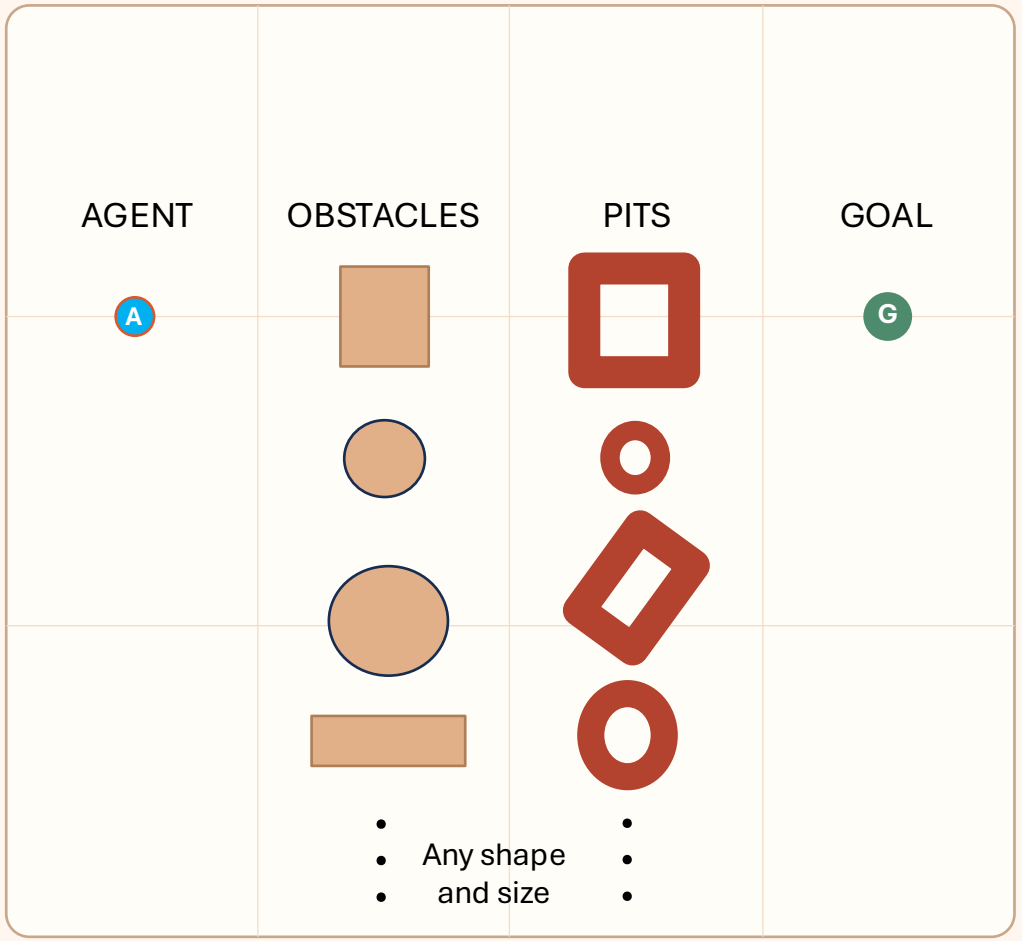
Oracle guidance
For robotics

The interesting question is not perception alone; it is instruction following under uncertainty.

TASK SETTING

Environment and task design

Each episode is a fresh random map, so the policy cannot memorize one fixed layout.



Bounded square world: random spawn, goal, obstacles, and pits

Episode mechanics

- World: bounded square map with random spawn and goal positions.
- Hazards: obstacles block motion; pits are terminal failures.
- Continuous control: normalized turn + forward action.
- Noise: linear, lateral, and angular slippage at execution time.

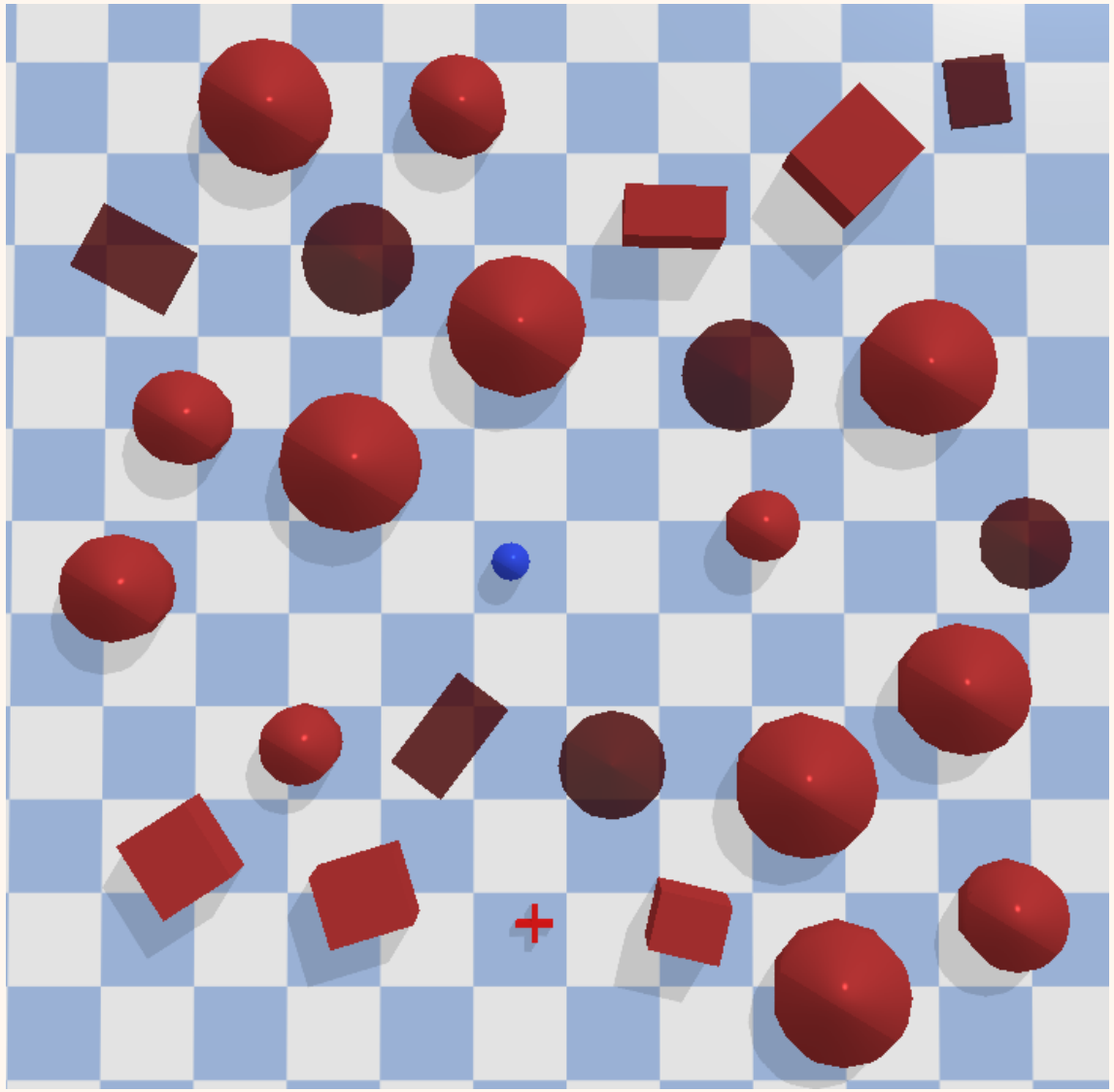
Why this is harder than standard navigation

- The goal is hidden from the policy.
- The agent must use memory between guidance refreshes.
- Local mistakes and catastrophic failures are both possible.

8–12	2–4	600	16
obstacles	pits	max steps	step advice cadence

Environment and task design

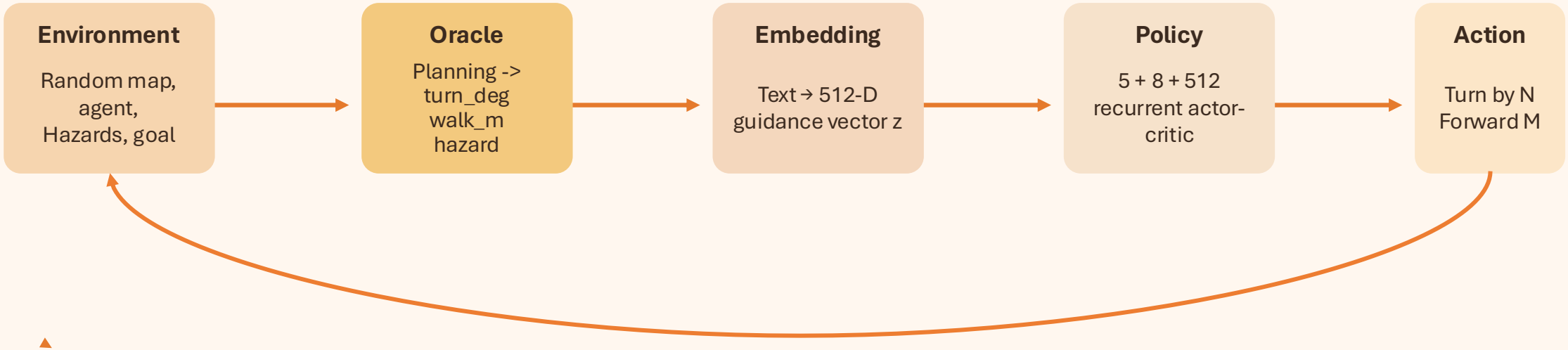
Screenshot of the actual pybullet enviroment with more hazards than normal



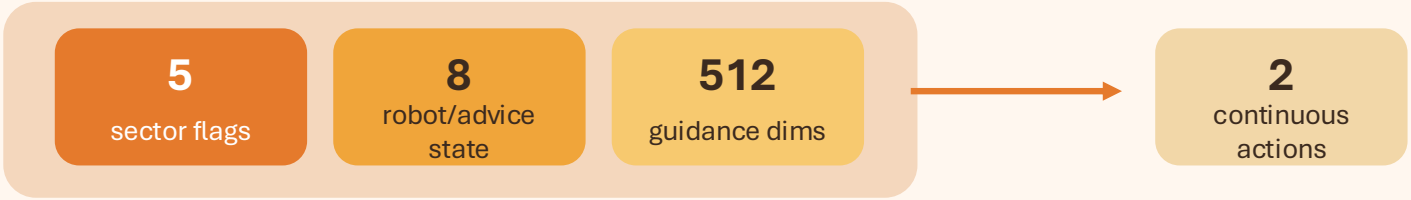
SYSTEM OVERVIEW

Full oracle-guided control loop

Full stack: environment design, oracle design, dataset engineering, model training, and evaluation tooling.



Loop intuition: the oracle provides high-level intent, while the policy must convert it into low-level robust execution over multiple noisy steps.



What the policy sees, decides, and optimizes

The design deliberately mixes tiny local sensing with a dense high-level conditioning vector.

Agent Observation

5 hazard sectors

8 robot state

512 guidance z

- Sector flags: front, front-left, left, front-right, right.
- Robot state includes x, y, heading sin, heading cos, advice age, and remaining instruction progress.
- The policy knows not only the advice, but where it currently stands relative to it.

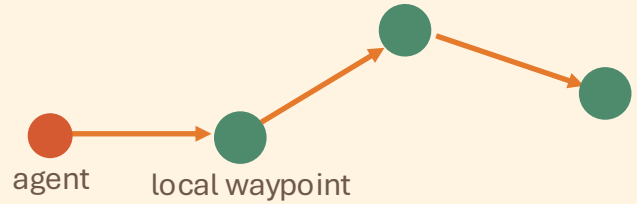
Action

2-D continuous output

turn ∈ [-1,1]
forward ∈ [-1,1]

- Turn is scaled by max_turn_deg.
- Forward is remapped from [-1,1] to [0,1].
- Execution noise makes the same command produce variable outcomes.

Reward shaping



- Per-step penalty, collision penalty, pit penalty, and final goal bonus.
- Most interesting (and necessary for training) choice: each oracle refresh defines a temporary local goal.
- The policy is rewarded for making coherent progress toward the current instruction.

This local-goal mechanism is the key bridge between abstract advice with sparse and rare reward and step-by-step control with good learning signal.

ORACLE LAYER

Two oracle modes, one shared interface

This abstraction is a strong engineering choice because both oracle paths end in the same 512-D conditioning signal.

Live LLM (openai gpt-5-mini API) oracle

- Receives the complete state of the enviroment.
- Receivesstrict prompt with frame conventions.
- Returns structured JSON and clamps the answer.
- Embeds final advice text into z ∈ R^512.

no raw free-form output, lower left/right ambiguity, same final interface as the dataset oracle. Model could work only with natural language, but no reason to not request the LLM for formatted output

Shared output contract

Advice fields

turn_deg • walk_m • hazard

Final representation

dense embedding z (512-D)

Dataset oracle

- Loads premade embeddings and corresponding advice.
- Matches candidates by turn/walk geometry and hazard filtering.

Limitation: heuristic controller for movement, not true semantic retrieval, so “not advanced thinking like LLM” .

- Advantage: much faster, basically necessary in training to avoid API latency.

Best practical choice for PPO training because it removes latency and cost from the loop.

OFFLINE DATASET

Premade advice dataset: the strongest systems component

.

```
embeddings= np.load("embedded_openai/embeddings.npy")
embeddings.shape
```

✓ 0.0s

```
(27972, 512)
```

Why the dataset matters

27,972 rows

512 embedding dim

219 embedding batches

531,695 tokens embedded

text-embedding-3-large embedding model

- Each row couples text advice with structured targets: turn angle, walk distance, hazard tag.
- Before training, the code checks dtype, shape, row alignment, and value normalization.
- Practical benefit: PPO can train without live API calls.

Two-stage learning strategy

Behavior cloning gives the guidance pathway a useful initialization before recurrent PPO takes over.

Stage A: behavior cloning

Input: guidance embedding z
Target: normalized turn + walk

→

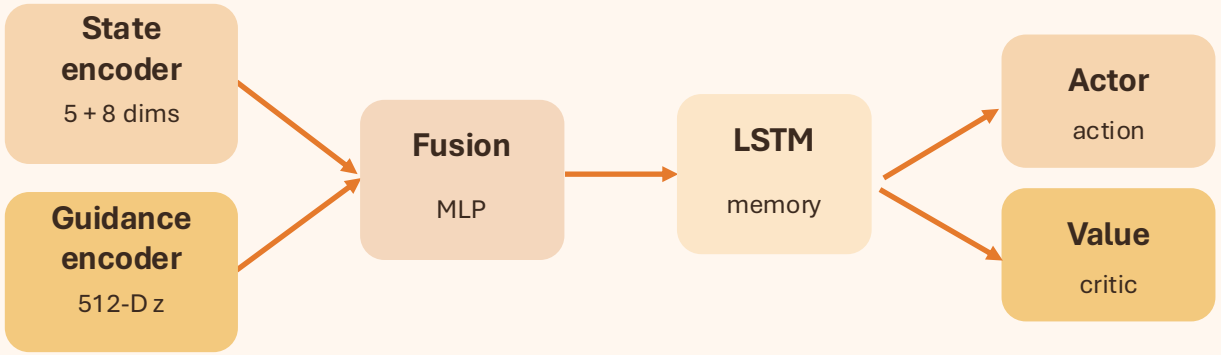
Transferred component

Guidance encoder weights
better PPO starting point

→

Stage B: recurrent PPO

Recurrent policy architecture



Why PPO needs help

- Random guidance processing at PPO start is unstable.
- The imitation loss keeps the policy aligned with current oracle intent.
- The LSTM handles delayed execution between refreshes.

BC does not solve navigation; it only bootstraps the guidance pathway.

Loss functions

Behaviour Cloning Loss

$$L_{BC} = \frac{1}{N}\sum_{i=1}^{N}[(\widehat{t_i} - t_i)^2 + (\widehat{w_i} - w_i)^2]$$

Terms

$\boldsymbol{t_i}$ = normalized turn targets
$\boldsymbol{w_i}$ = normalized walk target
$\boldsymbol{\widehat{t_i}}$ **and** $\boldsymbol{\widehat{w_i}}$ = model prediction from the 512-D advice embedding

PPO Loss

$$L_{PPO} = L_{clip} + \mathrm{L_{value}} - \mathrm{H}(\pi) + L_{imitation}$$

Terms

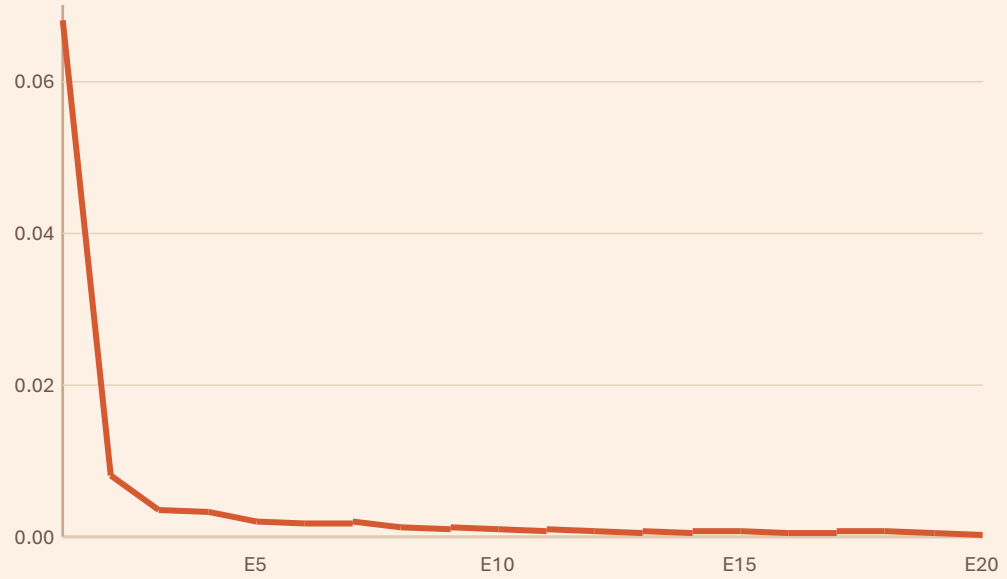
$\boldsymbol{L_{clip}}$ = Clipped ppo policy loss
$\boldsymbol{L_{value}}$ = critic/value loss
$\mathbf{H}(\boldsymbol{\pi})$ = entropy regularization
$\boldsymbol{L_{imitation}}$ = auxilliary imitation loss

TRAINING SIGNALS

What training suggests before held-out evaluation

The warm-start works very well offline, while PPO shows strong improvement on rolling training metrics.

Behavior cloning validation loss

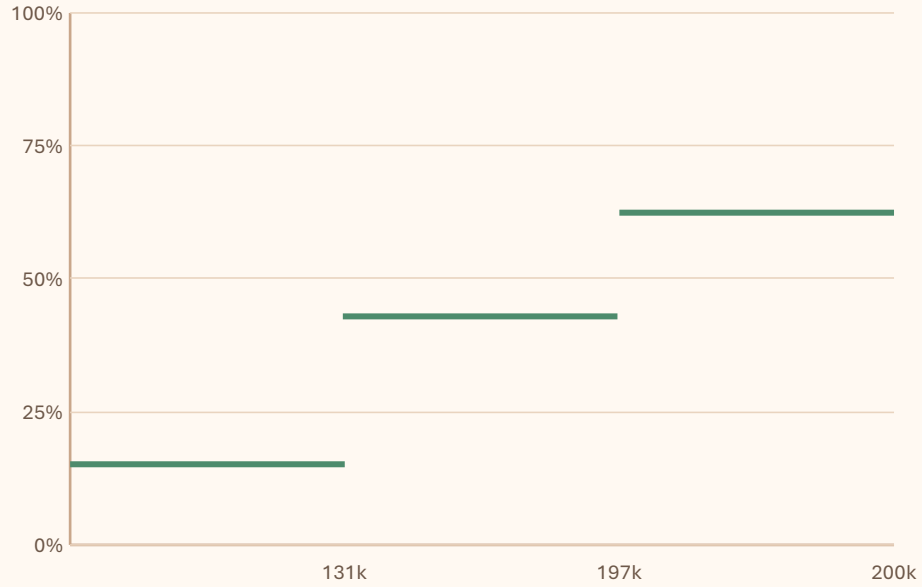


5.09e-4 best val loss

23,776 train rows

4,196 val rows

Rolling PPO success during training



200k env steps

1166 episodes

0.35 imitation coef

EVALUATION

Held-out evaluation

Held-out success rate

62%

Policy checkpoint details

-29.12 mean return	**11.13** Oracle calls per episode	**18%** pit fail
120.69 mean collisions		**62.6** steps to goal

Acceptance target was 70% success over 100 episodes. Current result: 62%, so acceptance is not met.

Bottom line: the bottleneck is execution learning, not just advice generation.

INTERPRETATION

What the results mean

.

What already works

•The guidance encoder tell us that the embeddings are expressive enough to be used as numerical instructions.
•The offline oracle and embedding pipeline are mature components.
•Oracle-only baselines provide a clean diagnostic reference.

This is stronger as a systems-and-methods project than as a final performance claim.

Where the policy still struggles

•Held-out success is below the 70% target.
•Collision count remains high, suggesting weak local avoidance.
•Agent can’t back away when having collided with obstacle
•The dataset oracle uses heuristics rather than semantic retrieval.

The likely gap is reliable execution under noise, hazards, and memory pressure.

Conclusion and next steps

Main takeaway: the contribution is the framework itself, plus clear evidence of what already works and what should improve next.

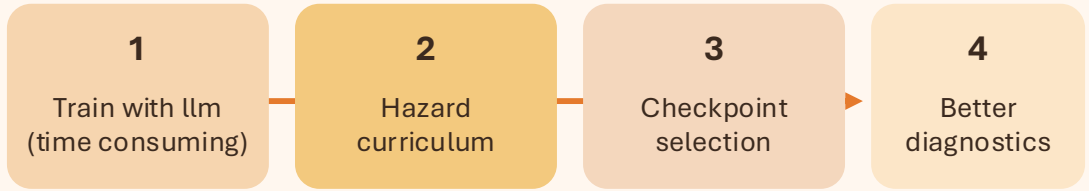
Takeaway

LLM advice can support RL training by extracting elements of the environment and aiding in task by acting as the brain and embedding models can out nuanced enough embeddings that can be mapped to actions

Most important contributions

- Unified live-vs-offline oracle abstraction.
- Balanced dataset and data-contract validation pipeline.
- BC warm-start + recurrent PPO training strategy.
- Oracle-only baseline for diagnosis and comparison.

Logical next steps



- Improve local obstacle avoidance to reduce the collision burden.
- Use richer failure traces to understand when memory breaks down.

Thank you

Any questions?