Extended Abstract

Motivation Long-horizon navigation among moving obstacles requires choosing where to be over time, not only the next dodge. A single policy that reads full state every step must solve planning and control together, which is difficult when episodes are long and rewards are sparse. This project asks whether hierarchical reinforcement learning helps when a high-level policy picks safe lane regions on a slower clock while a low-level policy handles per-step motion. The task is a custom 32×32 grid version of Asteroids: a bottom paddle must survive falling rocks as long as possible from compact vector observations rather than pixels.

Method A flat DQN baseline provides a strong reactive comparator, trained with shaped rewards and collision replay. The hierarchical stack has a manager that commits to one of twelve lane bins every $K = 16$ steps from a 42-dimensional scene summary: per-lane time-to-impact, cluster safety scores, and a transit mask for reachable lanes. A worker executes left, stay, or right each step from its own observation, which always includes the manager's current subgoal. Three worker variants are compared: reach-only (6-d, follows the target without rock coordinates), full sensing (69-d, subgoal plus all rock slots), and compact sensing (15-d, subgoal plus the three nearest rocks).

Implementation All agents share the same dodge physics and reward shaping. Workers are pretrained for their role—reach the subgoal or survive collisions—then frozen while the manager is trained with simulation-based lane planning. The manager ranks lanes from hand-designed hazard features rather than raw pixels. Evaluation uses 50 held-out episodes (seeds 0–49) with episode length as the primary metric.

Results HRL-compact achieves the longest mean survival (1031 ± 99 paddle steps), roughly $2.5\times$ the flat baseline (420 ± 56) and about $2\times$ HRL-full (515 ± 69). HRL-reach follows lanes most often (58% time in region) but dies quickly (143 ± 17 steps) because it cannot sidestep rocks that drift into its column. HRL-full beats flat with full rock sensing but collides on 98% of runs and rarely holds the manager's target (27% in region). Only HRL-compact reaches the 2,000-step cap on a substantial fraction of episodes ($22\% \pm 6\%$).

Discussion Hierarchy helps only when the worker is trained to dodge and given the right local view. Lane planning alone does not overcome a non-reactive worker; full rock state at every step can distract the worker from the manager's slower plan. The strongest design pairs manager lane commitments with a compact view of imminent threats rather than the entire rock field. Within these experiments, worker observation design and pretraining objective mattered more than further manager changes between the last two manager variants.

Conclusion This project shows that splitting planning from control can beat a tuned flat DQN in a vector-state dodge task, but only with complementary roles at each level. The central lesson is that hierarchy is only as strong as its worker: 15-dimensional observations (subgoal context plus three nearest rocks) achieve the best survival while using fewer inputs per step than either the flat agent or the full-sensing worker. Future work could tune the manager horizon, learn manager features end-to-end, and train both levels jointly across multiple training seeds.

Stay In Your Lane!
Hierarchical RL for a Modified Asteroids Game

Samantha Estrada
Department of Computer Science
Stanford University
estradas@stanford.edu

Abstract

Long-horizon dodge survival requires choosing where to be over time, not only the next reactive move. This report studies hierarchical reinforcement learning in a custom 32×32 asteroids environment with vector observations, comparing a tuned flat DQN to a two-level controller where a manager picks safe lane regions every 16 steps and a worker handles per-step motion. Three worker designs vary what the low level sees: subgoal only, full rock field, or subgoal plus three nearest rocks. The compact worker (15 dimensional) achieves the longest mean episode length (1031 ± 99 steps), roughly $2.5\times$ the flat baseline, even though it sees far less state than a worker with the full 69-dimensional rock view. Lane planning without dodging fails; full sensing helps but does not maximize survival. The results suggest that hierarchy works when each level has a clear job and the worker gets a small, task-relevant view of imminent hazards rather than the entire scene every step.

1 Introduction

Long-horizon navigation among moving obstacles requires choosing where to be over time, not only the next dodging maneuver. Tasks such as driving in traffic, avoiding debris, or tracking falling hazards share this structure: the agent must keep a safe position as threats move, rather than optimize one timestep at a time. A single policy that reads full environment state every step must handle both where to go and how to dodge immediately; such policies are expensive to train and struggle when rewards are sparse or episodes run long. Hierarchical reinforcement learning splits slow planning from fast control; this report investigates when that split improves survival and what each level needs to see.

This is studied in a small custom environment: a 32×32 grid where a paddle at the bottom must survive falling asteroids as long as possible. Physics are fixed, and the agent observes a compact vector of paddle and rock positions rather than raw pixels. The baseline is a flat DQN trained with shaped rewards and collision replay; it already dodges reactively and survives for hundreds of steps. The main question is whether a two-level controller can match or beat that baseline with a clearer split between planning and control.

The hierarchical design uses two roles: A manager picks one of 12 lane bins every $K = 16$ steps using a 42-dimensional map of the grid state: per-lane time-to-impact (TTI), cluster scores for which regions look safest, and a transit mask for lanes that can be reached without crossing danger. A fixed worker moves the paddle left, stay, or right each step from its own observation, which always includes the manager's current target (lane, time remaining in that target, and whether the paddle is already there). The manager plans lanes occasionally; the worker handles fine-grained motion every step.

I investigate three research questions.

RQ1: Does hierarchical lane planning (manager picks a bin every K steps; worker executes motion) improve episode survival versus the flat DQN baseline?

RQ2: What worker capability is required for the hierarchy to help: a compact subgoal only, detailed hazard sensing, or some combination of inputs?

RQ3: How should the manager–worker split over decision horizon, observations, and training be tuned for a compact but effective hierarchical policy?

Compared to the flat baseline, the hierarchical agent beats it on mean episode length only after the worker is trained for survival and given enough rock information to dodge effectively. An early version with a small non-reactive worker still dies quickly, even when the manager plans lanes and restricts moves to safe transit paths. What matters most is what the worker sees and how it is trained to stay alive; improving the manager alone does not close the gap. A more compact worker design (15 numbers: subgoal context plus the three closest rocks relative to the paddle) achieves the longest mean survival while using less input per step than the flat agent or a worker that sees the full rock field. Detailed metrics appear in the Results section.

2 Related Work

The flat baseline builds on deep Q-networks for Atari game control, where Q-learning with a neural network, experience replay, and a target network made it practical to learn from long episodes Mnih et al. (2015). The Arcade Learning Environment (ALE) remains the standard benchmark for game-playing agents Bellemare et al. (2013). This report uses the same DQN family—shaped rewards and replay—but studies a different task: ALE Asteroids rewards shooting and reads pixels, whereas the agent here dodges falling rocks from vector state on a custom grid. Prioritized and double Q-learning extend replay training on other Atari games Schaul et al. (2015); Van Hasselt et al. (2016); collision oversampling in this project serves a similar purpose but is simpler and tuned to this environment.

Hierarchical RL provides the motivation for splitting planning from control. The options framework models actions that run for multiple steps before ending Sutton et al. (1999), and MAXQ decomposes a task into subtasks with separate value functions Dietterich (2000); both describe a high-level policy that sets direction while a low-level policy handles motion step by step. The manager's 16-step lane commitments follow that pattern, but lane bins and target pockets are hand-designed rather than learned or discovered options.

The closest prior systems are deep manager–worker architectures for difficult games. h-DQN pairs a meta-controller that picks goals with a controller that acts every step Kulkarni et al. (2016), FeUdal Networks run a Manager at a slow time scale and a Worker at a fast time scale Vezhnevets et al. (2017), and Option-Critic learns options and their termination end-to-end on ALE games Bacon et al. (2017). These methods ask whether hierarchy helps in principle; this report asks a narrower diagnostic question—with manager features held fixed, what rock information and what pretraining does the worker need for lane planning to beat a strong flat DQN? Prior work typically learns high-level goals from pixels or latent vectors and trains both levels jointly, whereas here the manager reads engineered lane features (TTI, cluster score, transit) and the experiments compare three worker designs (6-d reach, 69-d full field, 15-d local rocks info only). Since the environment is a modified asteroids game, this report measures survival time on dodging tasks rather than score maximization.

3 Methods

3.1 Environment

All agents train and evaluate in the same dodge environment so comparisons reflect controller design, not different game rules. The playfield is a 32×32 grid. A three-column paddle sits on the bottom row and may move left, stay, or right one column per step. Up to twelve asteroids spawn from the top every four steps; each has a column, row, horizontal drift, and fall speed. An episode ends when an asteroid hits the paddle, or when a 2,000-step cap is reached. The agent does not receive pixels. The flat observation is a fixed 65-dimensional physics vector: the paddle's left edge position, plus sixteen rock slots (each storing column, row, drift, and fall speed; empty slots are zeros). This is the

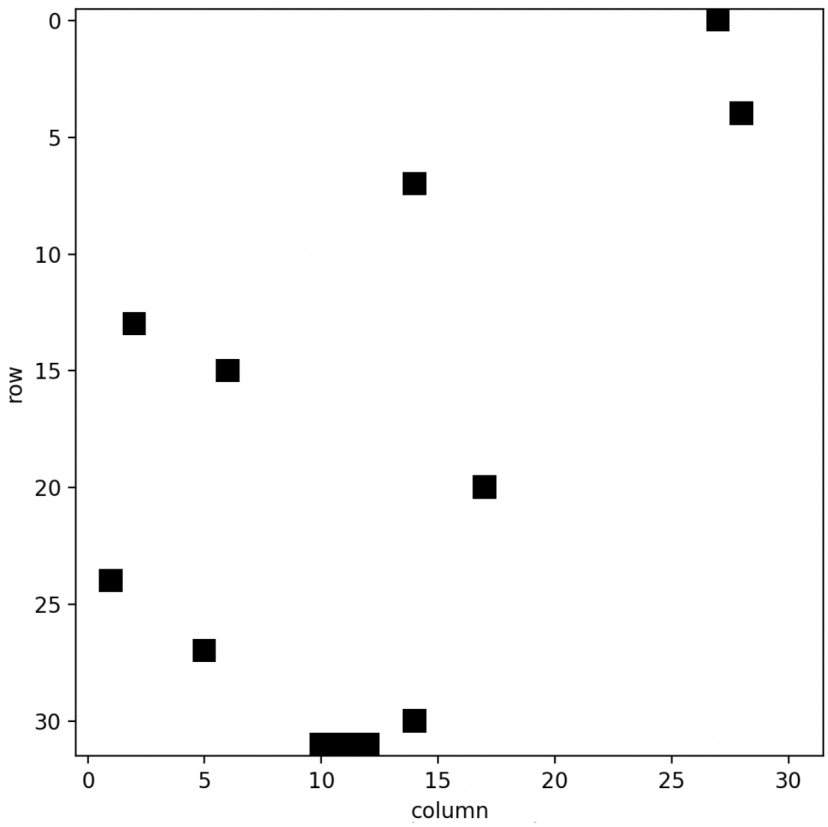

Figure 1: The 32×32 dodge environment. A paddle on the bottom row avoids asteroids that spawn from the top.

input to the flat DQN and to the HRL-full worker. Every agent optimizes the same per-step shaping reward. A surviving step earns a small alive bonus ($+0.01$). Clearance rewards open lateral space under low rocks (rocks with row > 24): $+0.02 \times \min(\text{gap}/16, 1)$, where gap is the nearest such rock to the paddle center. Proximity penalizes rocks in a danger zone six rows above the paddle and within four columns of center: $-\min(0.04 \times n, 0.15)$ for n rocks in that zone. A collision applies -1 and ends the episode. Full reward values and training hyperparameters are in the Experimental Setup and Appendix.

3.2 Flat baseline

Hierarchical methods are only meaningful if the flat baseline is strong. The baseline is a single DQN that reads the full 65-dimensional state every step and chooses left, stay, or right. This agent was tuned with clearance and proximity shaping, longer exploration decay, and collision replay (extra copies of collision transitions in the replay buffer). It learns reactive dodging and survives for hundreds of steps; Appendix A describes the tuning stages. This flat agent is the benchmark for all hierarchical stacks.

3.3 Hierarchical controller

The hierarchical agent separates where to be from how to move this step.

Manager. The manager acts every $K = 16$ environment steps. Because new rocks spawn every four steps, $K = 16$ spans about four spawn intervals—enough time for a falling rock to move a meaningful distance and for the paddle to reach a new lane target. Its action is a lane bin in $\{0, \ldots, 11\}$, which maps to a target pocket: the chosen bin plus its immediate neighbors on the

Table 1: Observation vectors for each policy level.

Policy	Dim	Contents
Flat DQN	65-d	Paddle left-x; 16 rock slots $\times$ (column, row, drift, fall speed).
Manager	42-d	Per-lane TTI, cluster score, transit flag (12 each); plus 6 context values.
HRL-reach worker	6-d	Normalized paddle x, target x, offset to target, dwell fraction, in-region flag, danger-zone count (no rock coordinates).
HRL-full worker	69-d	Full 65-d physics vector plus 4 subgoal dims (target x, offset, dwell, in-region).
HRL-compact worker	15-d	6-d compact vector plus top-3 urgent rocks $\times$ (relative Δcol, Δrow, fall speed).

bottom row (three bins total by default). Once chosen, that target stays fixed for 16 steps unless the episode ends. The manager does not see the raw 65-dimensional rock field. It sees a 42-dimensional planning vector with layout $[12\ \mathrm{TTI} \mid 12\ \mathrm{cluster} \mid 12\ \mathrm{transit} \mid 6\ \mathrm{context}]$:

- TTI (time-to-impact): for each lane bin, how many steps until an asteroid could hit the paddle if it stayed in that lane (normalized to $[0, 1]$ over a 20-step horizon). Low TTI means impact is soon.
- Cluster score: for each bin, the sum of capped TTI values over neighboring bins. High cluster score means a wider pocket of temporarily safe space, not just one empty column.
- Transit flag: for each bin, whether the paddle can move there from its current position without passing through immediately unsafe columns.
- Context (6 values): current lane bin, current target bin, dwell steps remaining, normalized paddle position, normalized danger-zone rock count, and TTI at the paddle's current position.

At each manager step, MaskedManagerDQN (the final manager version) zeroes out Q-values for illegal bins so the policy only selects transit-feasible targets or the current region.

Worker. The worker acts every environment step. It is pretrained separately, then frozen while the manager is trained on top. All workers receive the manager's subgoal context (target lane, steps remaining, in-region flag); what differs is how much rock information is included (Section 3.4). When the manager acts, the environment runs 16 worker steps before the next manager decision. The manager's reward is the sum of per-step shaping rewards over that block, so its return is not directly comparable to the flat agent's per-step return. We therefore compare agents mainly by episode length.

3.4 Observations

Table 1 summarizes every policy input. The flat agent and hierarchical workers read vectors built from the same underlying physics; the manager reads a compressed lane-level summary.

Flat and manager observations. The flat 65-d vector is the environment's native output: one paddle coordinate and up to sixteen rock entries (only twelve rocks may be active at once in the experiments run in this project). The manager's 42-d vector is computed from the same game snapshot but aggregates rocks into per-lane safety heuristics described above rather than listing individual positions.

Worker observations. Each hierarchical worker variant tests a different sensing–training tradeoff.

HRL-reach (6-d). *Input*: normalized paddle position; normalized target-lane position and offset; dwell time remaining (fraction of 16); binary in-region flag; and a normalized count of rocks in the proximity danger zone—but not where those rocks are. *Trained for:* staying inside the manager's target pocket (region-following reward), not long-horizon survival. *Tests whether:* lane planning alone is enough to survive.

Table 2: Shared DQN hyperparameters (all policies).

Parameter	Value
Learning rate	1×10^{-3}
Replay buffer size	150,000
Batch size	64
Discount γ	0.99
Exploration ε	$1.0 \rightarrow 0.05$ over 75% of training

Table 3: Training runs and checkpoint selection.

Component	Training steps	Best checkpoint by
Flat DQN	600k env	Eval reward
W2 worker (HRL-reach)	200k env	Time in region
W3 worker (HRL-full)	600k env	Episode length
W4 worker (HRL-compact)	600k env	Episode length
M3 manager	25k mgr	Eval reward
M4 manager	50k mgr	Episode length

HRL-full (69-d). *Input*: the full 65-d physics vector concatenated with four subgoal values (normalized target x, offset to target, dwell fraction, in-region flag)—the same rock view as the flat agent plus the manager's current assignment. *Trained for*: surviving as long as possible using full dodge shaping and collision replay; subgoal bonuses are small. *Tests whether:* flat-level sensing is enough for the hierarchy to beat the baseline.

HRL-compact (15-d). *Input*: the 6-d compact vector plus the three most urgent rocks, ranked by row (closest to paddle first) and then column distance. Each rock contributes three normalized values: horizontal offset from the paddle center, vertical distance above the paddle, and fall speed. Empty slots are zeros when fewer than three rocks are active. *Trained for:* surviving as long as possible, same approach as HRL-full. *Tests whether:* the worker needs the entire rock field, or only the most imminent nearby threats.

4 Experimental Setup

All policies use Stable-Baselines3 DQN (`MlpPolicy`) with learning rate 10^{-3}, replay buffer 150k, batch size 64, $\gamma = 0.99$, and ε decay from 1.0 to 0.05 over 75% of training (Table 2). Each checkpoint comes from one training run; reported uncertainty is over 50 held-out evaluation episodes.

4.1 Training

Workers and the flat agent train on every paddle step. The manager trains on high-level decisions every 16 paddle steps. Each worker is pretrained, then frozen while its manager is trained on top. Collision transitions are replayed $10\times$ during flat, worker, and manager training. Table 3 lists training length and how the best checkpoint was chosen.

4.2 Compared agents

Held-out evaluation compares four policies (Table 4). Flat DQN is a single policy with no hierarchy. Each hierarchical policy is a stack: one frozen worker paired with one manager trained on top. The stacks follow an experimental progression. HRL-reach (W2 + M3) is the earliest hierarchy: a reach-trained worker with no rock positions, paired with an M3 manager trained for 25k manager steps and selected by eval reward. It tests whether lane planning alone can work before investing in survival-trained workers. HRL-full and HRL-compact are later stacks that pair survival-trained workers (W3 or W4) with an M4 manager trained for 50k manager steps and selected by episode length during joint evaluation. M4 was introduced once workers could actually survive long enough for episode-length checkpointing to be meaningful. HRL-full and HRL-compact share the same M4

Table 4: Policies in held-out evaluation. Internal worker/manager IDs match training logs.

Agent	Stack	Worker	Manager	Worker obs
Flat DQN	—	—	—	65-d
HRL-reach	W2 + M3	W2	M3	6-d
HRL-full	W3 + M4	W3	M4	69-d
HRL-compact	W4 + M4	W4	M4	15-d

Table 5: Held-out evaluation (mean $\pm$ SE). Length is paddle steps.

Agent	Obs	Length	Reward	Collision	Cross-lane	In region
Flat DQN	65-d	420 ± 56	$+3.91 \pm 0.68$	$100\% \pm 0\%$	163 ± 24	—
HRL-reach	6-d	143 ± 17	-0.05 ± 0.16	$100\% \pm 0\%$	43 ± 5	$58\% \pm 3\%$
HRL-full	69-d	515 ± 69	$+2.32 \pm 0.57$	$98\% \pm 2\%$	107 ± 15	$27\% \pm 1\%$
HRL-compact	15-d	1031 ± 99	$+6.22 \pm 0.77$	$78\% \pm 6\%$	280 ± 27	$53\% \pm 1\%$

manager approach; they differ only in which frozen worker sits underneath, isolating the effect of worker observations.

4.3 Evaluation

Each policy is evaluated for 50 episodes with reset seeds 0–49, deterministic actions, and a 2,000-step cap. Results are mean $\pm$ standard error (SE) over those episodes. *Episode length (primary metric):* paddle steps survived before a collision or the time cap. *Collision rate:* fraction of episodes ending in contact with an asteroid. *Time in region (hierarchical only):* fraction of worker steps spent in the manager's target lane pocket. Reward totals follow the same ordering as length, but manager rewards sum over 16-step blocks and are not directly comparable to flat per-step rewards.

5 Results

5.1 Quantitative results

Table 5 and Figure 2 summarize held-out performance (50 episodes, seeds 0–49).

Four findings stand out:

1. HRL-compact survives longest (1031 ± 99 steps), roughly $2.5\times$ the flat baseline (420 ± 56) and about $2\times$ HRL-full (515 ± 69). Medians follow the same order (969, 342, 314, and 114 steps). All policies show high variance across seeds (Figure 2).
2. HRL-reach dies quickly (143 ± 17 steps) even though it follows the manager's lane most often ($58\% \pm 3\%$ time in region). Lane planning without rock-level dodging is not enough.
3. HRL-full beats flat but rarely finishes episodes. It collides on $98\% \pm 2\%$ of runs; only HRL-compact regularly reaches the 2,000-step cap ($22\% \pm 6\%$ of episodes).
4. More observations did not win. HRL-full sees the full 69-dimensional rock field; HRL-compact sees only 15 numbers (subgoal plus three nearest rocks) yet performs best. The right local features matter more than full state at every step.

HRL-compact also has the highest total cross-lane distance (280 ± 27 per episode), but this reflects longer episodes rather than more jittery movements per step: it simply has more time to move. Only HRL-compact reaches the 2,000-step time cap on a substantial fraction of seeds ($22\% \pm 6\%$); all other policies collide on nearly every run.

5.2 Qualitative results

Flat DQN reacts to the full rock field every step, shifting left or right to open gaps; it crosses lanes often and eventually collides despite reactive dodging. By contrast, HRL-reach tracks the manager's

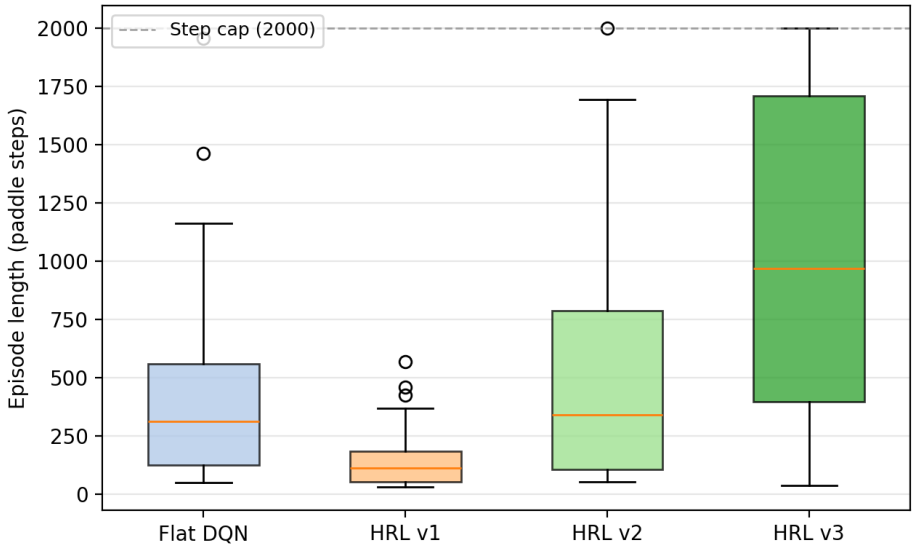

Figure 2: Episode length over 50 held-out episodes per policy. Dashed line: 2,000-step cap.

target pocket and often sits in the assigned region, but cannot see individual rock positions and does not sidestep rocks that drift into its column. HRL-full has the opposite problem: it dodges using the full 69-d rock view but frequently leaves the manager's target pocket to avoid imminent hits, consistent with its low time-in-region score ($27\% \pm 1\%$). HRL-compact balances both roles, following manager retargets while making smaller corrective moves based on the three nearest rocks; on favorable seeds it rides out long rock patterns and can reach the 2,000-step cap without contact.

6 Discussion

The first research question asks whether hierarchical lane planning improves survival over a strong flat DQN. The answer is conditional: planning alone does not. HRL-reach survives far less than flat (143 vs. 420 mean steps) despite spending the most time in the manager's target region. Once the worker is trained for survival and given rock-level information, the hierarchy can surpass the baseline. HRL-compact more than doubles flat mean episode length and is the only stack that regularly reaches the 2,000-step cap.

The second question asks what worker capability the hierarchy needs. Subgoal following alone is insufficient: without rock coordinates, the worker cannot sidestep threats that drift into its lane. Full rock sensing helps (HRL-full beats flat) but is not the best design here. The strongest stack combines the manager's lane target with a small view of the three most urgent rocks (15 numbers total). Survival training also matters: the reach-trained worker was never meant to dodge, and pairing it with a capable manager does not recover performance.

The third question concerns tuning the manager–worker split. This report fixes $K = 16$ and hand-designed manager features, and holds the manager strategy fixed while varying only the worker. Within that scope, the most impactful knob was worker observation design and pretraining objective, not further manager changes between M3 and M4. A fuller answer to RQ3 would require sweeps over K, manager feature sets, and joint training rather than a frozen worker; that remains open.

6.1 Why HRL-compact outperforms HRL-full

HRL-full sees strictly more information than HRL-compact (69-d vs. 15-d), yet survives roughly half as long on average. One explanation is division of labor. The manager already summarizes where open lanes are; the worker's job is fine-grained survival inside that plan. The three nearest rocks are the signals most relevant to immediate collision avoidance. The full 16-slot rock field may add

noise and make it easier for the worker to override the manager's lane commitment, which matches HRL-full's low time-in-region score (27%) despite longer episodes than the flat baseline.

HRL-compact sits in a middle ground: enough local detail to dodge, few enough inputs to learn a stable policy, and enough lane compliance (53% time in region) to benefit from the manager's slower plan. High time in region does not guarantee safety (HRL-reach proves that) but some balance between following the manager and breaking away to dodge appears important.

6.2 Limitations

All reported checkpoints come from a single training run per component; uncertainty is over 50 evaluation episodes, not over training seeds. Episode lengths vary widely across seeds (Figure 2), so mean performance should be read with that variance in mind. The manager relies on hand-designed TTI, cluster, and transit features rather than learned abstractions from pixels or raw state. Workers are pretrained and then frozen during manager training, which simplifies the experiment but may leave performance on the table if the two levels could adapt together. Finally, results are specific to this grid environment with vector observations; transfer to visual inputs or different physics is untested.

7 Conclusion

This project studied hierarchical reinforcement learning for long-horizon dodge survival in a 32×32 asteroids environment. A tuned flat DQN is a capable reactive baseline, but a two-level controller can beat it when the levels are given complementary roles: the manager commits to safe lane regions every 16 steps, and the worker handles per-step motion using a compact view of imminent threats.

The central lesson is that hierarchy is only as strong as its worker. Lane planning without dodging fails; dodging with the full rock field can help but does not maximize survival here. The best policy uses 15-dimensional worker observations—subgoal context plus three nearest rocks—and achieves the longest mean episode length while using fewer inputs per step than either the flat agent or the full-sensing worker. Future work could tune the manager decision interval, learn manager features end-to-end, train manager and worker jointly, and repeat the comparison across multiple training seeds to tighten uncertainty estimates.

8 Team Contributions

This was a one person project—all work and experiments were designed and conducted by me.

9 AI Tools Disclosure

I used AI coding assistants (e.g., Cursor) for boilerplate training and eval scripts, Stable-Baselines3 DQN setup and API usage, plotting data, and debugging. I designed the environment (including hyperparameters), hierarchical manager/worker setup, rewards, worker observations, experiments, and training runs on my own.

Changes from Proposal The original proposal focused on learning a latent world model from 32×32 pixel Pong under partial observability, with velocity hidden from the agent. I shifted to a multi-object Asteroids-style dodge game with vector physics observations, deferring pixel-based dynamics modeling to future work. This followed the proposal's own concern that Pong might be too easy for advanced methods, and flat experiments here showed that many moving rocks create a clearer need for strategic positioning. The main research direction became hierarchical RL—a manager for lane planning and a worker for local control—with experiments over worker observations rather than comparisons to frame-stacked or CNN pixel baselines.

References

Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research* 47 (2013), 253–279.

Thomas G Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of artificial intelligence research* 13 (2000), 227–303.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems* 29 (2016).

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).

Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112, 1-2 (1999), 181–211.

Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*. PMLR, 3540–3549.

A Flat Baseline Development (Summary from Milestone)

Before hierarchical training, the flat DQN was improved through staged reward and training changes:

1. Sparse reward (-1 on hit only): the agent camped in a corner; no real dodging.
2. + Alive bonus ($+0.01$/step): longer survival, but still returned to a home column.
3. + Clearance and proximity shaping (Table 6): pushed the agent toward open lanes.
4. + DQN tuning: longer ε decay, 150k replay buffer, collision replay $10\times$, eval every 5k steps over 600k total steps.

Figure 3 shows eval reward and episode length during this run. The best checkpoint from this run is the flat comparator in all later experiments.

Table 6: Full shaping reward (per paddle step, if no collision).

Term	Value
Alive bonus	$+0.01$
Clearance	$+0.02 \times \min(\text{gap}/16,\ 1)$
Proximity	$-\min(0.04 \times n,\ 0.15)$, n rocks in danger zone
Collision	-1 (episode ends)

Figure 3: Flat DQN eval reward and episode length during training (600k paddle steps).