Phase 3 – Hierarchical Reinforcement Learning & Curriculum Training

Project: *Autonomous Parking – 2D + 3D Simulation System*
Author: Vishwaksena Dingari
UID: 121113764
Course: MSML 642 – Final Project
Date: November 24, 2025

1. Overview

Phase 3 is where the project moves from basic kinematic control (Phase 2) to a full **Hierarchical Reinforcement Learning (HRL)** setup.
The goal is to train an agent that can:

- Navigate non-trivial parking lots using a global route, and
- Execute a precise final parking maneuver into a target bay.

This phase introduces:

1. **Hierarchical Architecture**: A* as a global planner + PPO as a local controller.
2. **Advanced Environment**: Dense path generation, spline smoothing, smart waypoint subsampling, and distance-normalized rewards.
3. **Micro-Curriculum Learning**: A 15-stage progressive training schedule.
4. **Training Pipeline**: Integrated video recording, checkpointing, and evaluation callbacks.

This report describes the main components implemented in Phase 3 and how they build on the work done in Phase 1 and Phase 2.

2. System Architecture

The system uses a **hierarchical control strategy** with two clear levels:

- A **high-level global planner** that plans a collision-free path using A*.
- A **low-level PPO policy** that follows waypoints and performs the actual parking.

2.1 Global Planner (High-Level)

- **Algorithm**: Grid-based A* search.
- **Resolution**: 0.25 m grid.
- **Input**:
  - Static occupancy map (walls and obstacles).
  - Target parking bay (goal pose).
- **Output**:
  - A collision-free sequence of points from spawn pose to the bay.

Post-Processing Steps

After A* generates a raw path, several steps are applied:

1. **B-Spline Smoothing**
   - The discrete A* path is passed to `scipy.interpolate.splprep`.
   - A smoothing parameter `s = 0.1` is used to get a tight but smooth curve.
   - This removes unnecessary zig-zags and produces a continuous path.
2. **Goal Alignment / Phantom Point Removal**
   - Earlier, the path sometimes ended at a "phantom" point slightly beyond or off the true bay center.
   - The spline is now sampled carefully so the final point lies exactly at the goal bay center.
   - Any extra terminal point is removed so the final waypoint is geometrically accurate.
3. **Smart Subsampling**
   - The dense spline path is reduced to a **sparse set of key waypoints**:
     - Points with sharp heading changes (e.g. > 15°) are always kept.
     - A minimum spacing (e.g. 2.5 m) is enforced between consecutive waypoints.
     - The total number of waypoints is capped (e.g. max 18) to keep the RL problem manageable.
   - These waypoints are fed to the PPO agent one by one as targets.

2.2 Local Planner (Low-Level PPO Controller)

- **Algorithm**: Proximal Policy Optimization (PPO).
- **Role**: Track the current global waypoint and complete the final parking manoeuvre.
- **Input**: A 43-dimensional normalized observation vector.
- **Output**: Continuous action `[speed, steering]`.

The PPO policy runs at each time step and decides how to steer and accelerate the vehicle to reduce the error to the current waypoint and the final bay pose.

3. Environment Implementation

The main training environment (e.g. `waypoint_env.py`) was extended and cleaned up in Phase 3 to support stable HRL training.

3.1 Observation Space (43D)

The agent receives a **43-dimensional vector**, including:

1. **Waypoint Information**
   - Relative position to the current target waypoint: `(dx, dy)` in the vehicle frame.
   - Relative heading difference to the waypoint: `dtheta`.
2. **Vehicle State**
   - Longitudinal velocity `v`.
   - Steering angle `delta`.
3. **Goal Bay Information**
   - Relative position to the bay center: `(cx, cy)` in the vehicle frame.
   - Yaw error `yaw_err` between vehicle heading and bay orientation.
4. **Lidar**
   - 32-ray simulated lidar distances around the vehicle to capture nearby obstacles.
5. **Progress Indicators**
   - `waypoint_progress` $\in$ [0, 1]: fraction of waypoints completed.
   - `is_near_goal` (0/1): flag indicating that the vehicle is close to the final bay.

Normalization:
All features are scaled either to `[-1, 1]` or `[0, 1]` (depending on their nature) to help the neural network train more reliably and avoid exploding gradients.

3.2 Action Space

The PPO policy outputs two continuous values:

- **Speed command** ∈ `[-1, 1]`
  - Mapped internally to a physical speed range `[-max_speed, max_speed]` .
- **Steering command** ∈ `[-1, 1]`
  - Mapped internally to steering angle range `[-max_steer, max_steer]` .

This allows the network to work in a normalized space while the environment converts actions to meaningful vehicle controls.

3.3 Reward Function Design

Parking is naturally a sparse-reward problem (success only at the end), so a shaped, dense reward function was implemented.

A. Distance-Normalized Waypoint Bonus

Problem:
Longer paths produced more total waypoint bonuses, which biased the agent towards long routes.

Solution:
Introduce a fixed `REWARD_BUDGET` for the whole path and distribute it by segment length.

- Let:
  - `total_length` = total length of the A* / spline path.
  - `segment_length` = length of a given path segment between two waypoints.
  - `BUDGET = 300.0` .
- For each segment, the bonus is:

$$
\text{bonus} = \text{segment_length} \times \left(\frac{BUDGET}{\text{total_length}}\right) \times \text{progressive_multiplier}
$$

- `progressive_multiplier` slightly increases bonuses for later path segments to encourage completing the whole path.

Result:
Agents following a short path and a long path obtain comparable total rewards for "doing the right thing," independent of spawn distance.

B. Phase-Blended Rewards

Reward weights are blended based on the current "phase" of the episode:

1. **Navigation Phase**
   - Early in the episode, the main objective is to reach waypoints in order.
   - Reward focuses on reducing distance to the current waypoint and making progress along the path.
2. **Approach Phase**
   - When near the bay but not yet properly aligned.
   - Reward balances:
     - Remaining waypoints, and
     - Decreasing `cx, cy` and `yaw_err` with respect to the bay.
3. **Parking Phase**
   - When the vehicle is close to the bay and near the final few waypoints.
   - Reward strongly emphasizes pose accuracy inside the bay:
     - small `|cx|`, `|cy|`, and `|yaw_err|`.

This phase-aware blending helps the agent shift focus from global navigation to fine parking alignment.

C. Penalties and Safety Terms

To avoid bad behavior:

- **Collision penalty**: e.g. `-20.0` and immediate episode termination.
- **Anti-Freeze**: about `-0.3` when the vehicle is "stuck" (low speed, not reducing goal error).
- **Time penalty**: small negative value per step to encourage faster completion and avoid unnecessary looping.

These terms prevent the agent from abusing the reward function and promote efficient, safe behavior.

3.4 Path Generation Pipeline

The path generation pipeline for each episode is:

1. **Dense Path from A***
   - Run A* on the occupancy grid to get a dense, collision-free path.
2. **Spline Smoothing**
   - Use B-spline fitting to produce a smooth continuous curve passing near the A* points.
3. **Smart Subsampling**
   - Keep points where turning curvature is high (heading change > 15°).
   - Enforce minimum spatial spacing, e.g. 2.5 m.
   - Cap maximum number of waypoints (e.g. 18).
4. **Exact Goal Reaching**
   - Sample spline over the full parameter interval `[0, 1]` .
   - Ensure the last sampled point is exactly at the bay center.
   - Remove any extra "phantom" end point.

This makes the path easy to follow for the PPO policy but still accurate with respect to the environment geometry.

4. Curriculum Learning (`curriculum.py`)

To manage the difficulty of the parking task, a **15-stage micro-curriculum** was implemented.

4.1 Stages (S1–S15)

The stages are grouped into five phases:

- **Phase 1 (S1–S3)**
  - Single bay, aligned start pose.
  - Increasing distance from the bay.
- **Phase 2 (S4–S6)**
  - Multiple bays in the same general orientation.
  - Forces the agent to handle slightly more variation.
- **Phase 3 (S7–S9)**

    - Perpendicular parking (approx. 90° turns).
    - The agent must learn to swing into the bay from sideways approaches.
- **Phase 4 (S10–S12)**
    - All four orientations (North, South, East, West).
    - Covers a full range of bay orientations and approach angles.
- **Phase 5 (S13–S15)**
    - Multi-lot generalization (e.g. Lot A and Lot B layouts).
    - Tests whether the agent can generalize to different topologies.

Each stage defines spawn distributions, available bays, and orientations.

4.2 Advancement Logic

The agent moves to the next stage when:

1. A **minimum number of steps** (or episodes) in the current stage has been completed.
2. Optionally, a **rolling success rate threshold** is satisfied (e.g. a small but positive success rate).

This prevents the curriculum from advancing too quickly before the policy has made reasonable progress on the current stage.

4.3 Replay Mechanism

To tackle **catastrophic forgetting**:

- Episodes from previous stages are sampled with probability `replay_prob`.
- This probability gradually increases from 0.0 up to around 0.7 as training goes on.

This ensures the agent continues to see earlier, simpler tasks and does not forget how to solve them while learning later stages.

5. Training Pipeline (`sb3_train_hierarchical.py`)

The training script uses `stable_baselines3` PPO with custom wrappers and callbacks to support curriculum and logging.

5.1 PPO Hyperparameters

Key settings:

- **Learning Rate**: `1e-4`
  - Lowered for more stable, fine-grained updates.
- **Batch Size**: `64`
- **Gamma (Discount Factor)**: `0.99`
- **Entropy Coefficient (`ent_coef`)**: `0.05`
  - Encourages exploration.
- **Clip Range**: `0.1`
  - Conservative updates to avoid destabilizing the policy.

These values were chosen based on early experiments and standard PPO practice for continuous control.

5.2 Callbacks

The following callbacks are integrated:

1. **VideoRecorderCallback**
   - Records selected training episodes as MP4 files.
   - Overlays the A* path and current trajectory.
   - Configured to work smoothly in the current OS environment.
2. **CurriculumEarlyStopCallback**
   - Monitors rolling success rates and other metrics.
   - Logs curriculum stage transitions to TensorBoard.
   - Can be used to decide if a stage is "good enough" to move on.
3. **CheckpointCallback**
   - Saves model checkpoints every fixed number of steps (e.g. every 100k steps).
   - Allows resuming training from intermediate points and comparing different stages.
4. **EvalCallback**
   - Periodically evaluates the policy on separate evaluation scenarios.
   - Records evaluation metrics independent of the training environment.

Together, these callbacks give visibility into training progress and allow safe, incremental development.

6. Critical Fixes and Improvements

During Phase 3, several issues were discovered and fixed:

1. **Crash at Final Waypoint**
   - An `IndexError` occurred when the agent reached the last waypoint and the code attempted to access the next one.
   - The indexing logic was updated to correctly detect the terminal waypoint.
2. **Double Success Tracking**
   - In `reset()`, success counts were sometimes updated twice, leading to incorrect success statistics.
   - The bookkeeping was cleaned up so each successful episode updates the counters exactly once.
3. **Phantom Point Removal**
   - The previous path smoothing sometimes appended an extra point that did not correspond exactly to the bay center.
   - The spline sampling and post-processing were revised to:
     - Sample over the correct parameter range.
     - Remove any nonessential tail point.
     - Guarantee that the final point is the bay center.
4. **Edge-Case Safety for Short Paths**
   - In rare cases where A* returned extremely short paths (e.g. 0 or 1 waypoint), some assumptions in the downstream code broke.
   - Additional guards were added to handle these edge cases gracefully and fall back to safe defaults.

These fixes improved stability and removed several sources of rare but severe training failures.

7. Conclusion and Next Steps

Phase 3 sets up a complete **hierarchical RL framework** for autonomous parking:

- Global A* planner with spline smoothing and subsampled waypoints.
- Local PPO controller operating on a 43D normalized observation space.
- Dense, phase-aware reward shaping with distance-normalized waypoint bonuses.
- A structured 15-stage curriculum with replay to avoid forgetting.

- A training pipeline with logging, video recording, evaluation, and checkpoints.

Current Status:

- Environment and planner integrated.
- PPO training with curriculum is running.
- Codebase has been debugged to remove major stability issues.
- The project is at the "80% complete" level expected for Phase 3.

Next Steps:

1. Run long training sessions (e.g. up to several million steps).
2. Analyze training and evaluation metrics for success rates and alignment quality.
3. Tune hyperparameters and reward weights if needed.
4. Prepare final demos and documentation for the full project submission.