POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

PHANTOM: A Hierarchical Learning Process for Agricultural Navigation

Laurea Magistrale in Automation and Control Engineering - Ingegneria dell'Automazione e del Controllo

Author: Davide Jarik De Rosa

Advisor: Prof. Matteo Matteucci

Co-advisor: Marcelo Becker

Academic year: 2025-2026

1. Introduction

Autonomous navigation of agricultural robots has remained a challenging and open problem in recent years. Crop-row navigation with wheeled robots is fundamental for planting, weeding, harvesting and high-throughput phenotyping. To be deployable at scale, agricultural rovers must be compact and low-cost, and operate reliably under the crop canopy, where global positioning is unreliable, perception is heavily occluded, and crops, terrain and weather vary continuously.

The majority of current methods can be classified in two families of approaches: classical separation of perception and control modules and direct end-to-end reinforcement learning. In the first class, a perception module estimates a compact, human-interpretable representation, typically crop-row or line parameters, which is then fed to a model-based controller. In this design some information is inevitably lost at the interface between the two modules, and the controller depends on an inevitably imprecise dynamic model and on an inaccurate state estimate. On the other hand, end-to-end approaches remove the interface by mapping perceptual observations directly to navigation actions. This requires the usage of reward functions that are not very simple or intuitive to implement, leading to hard-to-deploy policies.

This thesis proposes **PHANTOM**: **P**re-trained **H**ierarchical **A**gricultural **N**avigation **T**rajectory **O**riented **M**odel, a two-stage hierarchical learning process that reconciles the two paradigms. The core idea is to first train a controller by shaping expert knowledge into a structured latent space using privileged information, and then reuse part of that frozen module with exteroceptive information to solve distinct navigation tasks. The hierarchy trains the two modules separately without losing information at their interface, and the pre-trained controller allows for simple and robust reward functions. The full system architecture is summarized in Figure 1. The main contributions of this work are:

- A scalable, pre-trainable network architecture for agricultural navigation enrichable by real-world expert demonstrations.
- A two-stage hierarchical learning process fully validated in simulation, with some real-world field experiments.

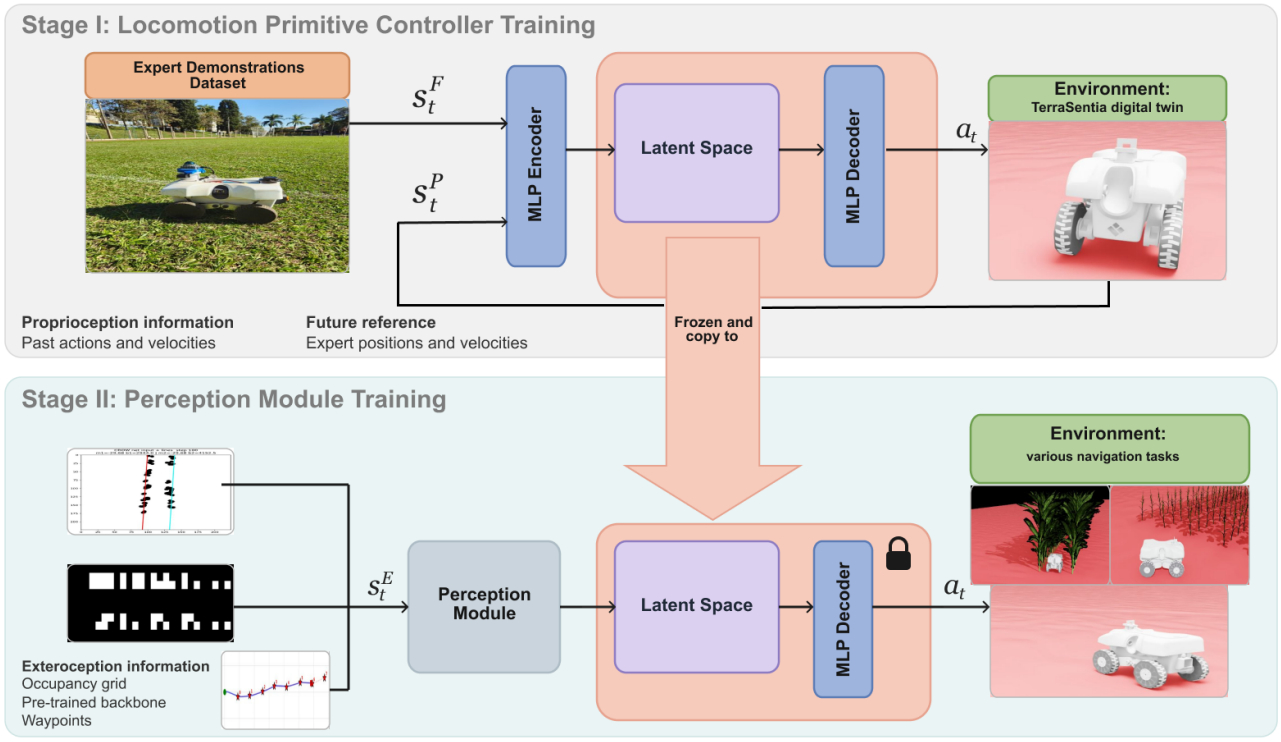

Figure 1: A framework overview of the proposed method

- An extended trainable virtual environment for the TerraSentia robot fully implemented in the Isaac Lab software.

2. Materials and Methods

2.1. The Robot Platform

The Phantom architecture is validated on the TerraSentia platform, a light-weight, compact four-wheel robot commercialized by EarthSense. All the wheels are actuated independently on the low-level by the only accessible control inputs: the linear and angular body velocities (v, ω). The important sensors for this work are a Bosch BNO055 IMU (Inertial Measurement Unit), a Zed-F9P GPS (Global Positioning System), and a Livox Mid-360 3D LiDAR, all connected to a Raspberry Pi 3.

2.2. Stage I: Controller Training

In the first stage, a dataset of expert demonstrations is collected from two kinds of experts: a nonlinear model predictive controller (NMPC), with several parameter configurations, and a heuristic algorithm. The NMPC implements a waypoint following navigation configured to mimic the under-canopy navigation, while the heuristic algorithm performs low linear velocity and high angular velocity turns to simulate a change of row over-canopy.

The expert knowledge from the final dataset is compressed with knowledge enhancement into a structured latent space by using the Imitation Learning (IL) paradigm. To compress knowledge from different experts into a single latent space, we adopt a control-oriented Variational Autoencoder (VAE) [2] architecture, that we call Locomotion Primitive Controller (LPC), and a discrete, vector-quantized counterpart (VQ-LPC) based on the Vector-Quantized Variational Autoencoder (VQ-VAE) [4]. In the Stage I of training, an encoder constructed by MLPs maps the proprioceptive state s_t^P (past actions and body velocities of the digital twin in Isaac Lab) and a future reference s_t^F extracted from the dataset expert trajectory to the latent space. Later, a decoder, also constructed by MLPs, generates an action a_t based on the selected latent instance. The training objective is to minimize a loss function composed by a reconstruction term and a latent one. While the latent loss is the standard for either VAE or VQ-VAE, the reconstruction loss in this case is the normalized error between the controller action a_t and the expert action a_t^*:

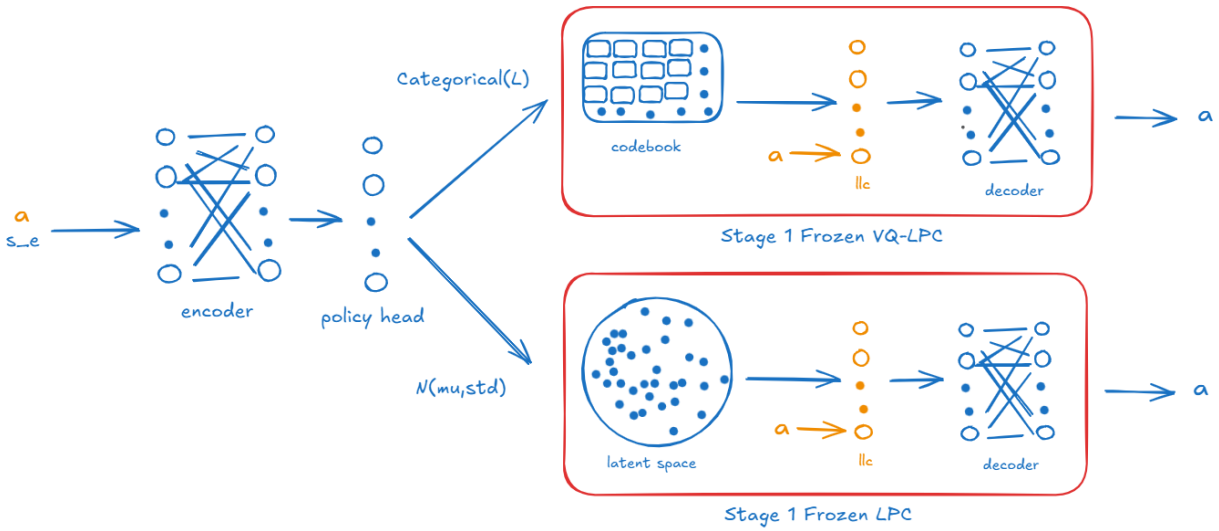

Figure 2: Stage II full network architecture

$$\mathcal{L}_{LPC} = \|a_t - a_t^*\|^2 + \\ - \frac{\beta}{2} \sum_{j=1}^{M} \left(1 + \ln \sigma_j^2 - \mu_j^2 - \sigma_j^2\right) \quad (1)$$

$$\mathcal{L}_{VQ\text{-}LPC} = \|a_t - a_t^*\|^2 + \|\text{sg}[\boldsymbol{z}_e] - \boldsymbol{e}\|_2^2 + \\ + \beta_c \|\boldsymbol{z}_e - \text{sg}[\boldsymbol{e}]\|_2^2 \quad (2)$$

where β and β_c are latent coefficients used to weight the losses and sg$[\cdot]$ is the stop-gradient operator. Additionally, some state-conditioned controller variants are tested. In those variants, the decoder is conditioned on both the latent vector and the last action, $p_\theta(a_t \mid \boldsymbol{z}_t, a_{t-1})$, through a low-level controller module (LLC) that balances the contribution of the latent vector and of the past action. The training process happens in our simulated environment. In the environment, each agent is spawned at a random position and orientation, the wheel friction coefficient is randomized, a random constant force and sudden impulse torques are applied for each episode. A random episode is chosen for each agent from the dataset, where the episode starts at the frame where the first non-zero action from the expert is applied. The network receives its inputs centered in the robot reference frame, and the episode stops and another one is selected when there are no more available privileged information from the dataset. Usually in IL problems where the agent tries to learn directly the action distribution of the expert (Behavior Cloning), a roll-out dataset aggregation is needed. In our case, we show empirically by monitoring the rate of out-of-distribution (OOD) actions of the models in the Stage II that in our setting this is not required.

2.3. Stage II: Perception Training

Stage II of training is summarized in Figure 2: the Stage I encoder is discarded, while the decoder (and, for the VQ-LPC variant, the codebook) is loaded and frozen, becoming the action space of a perception policy (a new encoder) trained by Proximal Policy Optimization (PPO) [5]. In PPO, an actor-critic method, a policy and a value function are optimized simultaneously. The loss function is a combination of three terms: the surrogate loss, the value function loss, and the entropy loss

$$L^{PPO} = \hat{\mathbb{E}}_t \left[L_t^{CLIP} - c_1 L_t^{VF} + c_2 S[\pi_\theta] \right], \quad (3)$$

where c_1, c_2 are coefficients for balancing the terms. The new encoder maps the exteroceptive observation s_t^E, and in case of state-dependent controller also the last action, to the latent space by means of a policy head composed of MLPs. For the LPC, the policy head outputs a diagonal Gaussian over the continuous latent space; for the VQ-LPC a categorical distribution over the frozen codewords. s_t^E can be, depending on the task and the implementation, a robot-centric Li-

DAR occupancy grid, a feature vector of a fine-tuned perception backbone, or a waypoint in the robot reference frame.
Three tasks are performed with the controllers trained in Stage I: waypoint-following, under-canopy navigation, row-change over-canopy. For the first task, the domain randomization techniques presented in Stage I are re-used. Each episode starts with the agent in a random position and orientation and the first waypoint is spawned. When the agent gets close enough to the waypoint, it obtains a reward and a new waypoint is spawned. An episode terminates either with a time-out after 20s (success), or when the agent does not reach the waypoint in time (fail). The reward function is simple and composed by only sparse rewards:

$$R_{wp} = w_g r^{\mathrm{g}} + w_t r^{\mathrm{t}}, \tag{4}$$

where w_g and w_t determine the weight respectively of reaching the waypoint and a fail. For the two agricultural navigation tasks, the randomization happens with noise on the LiDAR readings, random position and orientation of the plantation rows, random (unperceivable by the agent) obstacle diameter, range of agent's spawning position and orientation. In this case an episode terminates either positively (reached the goal position) or negatively (collision or time-out). The reward functions are again simple:

$$R_{uc} = w_g r^{\mathrm{g}} + w_c r^{\mathrm{c}} + w_t r^{\mathrm{t}}, \tag{5}$$
$$R_{rc} = w_a r^{\mathrm{a}} + w_g r^{\mathrm{g}} + w_c r^{\mathrm{c}}, \tag{6}$$

where w_g, w_t, w_c are sparse reward coefficients which weight, respectively, goal reached, time-out, and collision. The only dense reward used is in row-change task, and it is intended to penalize unexpected actions for a specific task.

3. Results and Discussion

3.1. Simulation results

The pipeline for simulation results was divided into two stages: we first trained different controllers from a single NMPC expert and evaluated on under-canopy navigation, we then selected the two best controllers and re-evaluated them on the full pipeline (multiple experts, three navigation tasks).

The tested controllers were a LPC, a VQ-LPC, and their state-dependent variants. All those reached high success rates, reasonable OOD-action rates, and were able to perform the under-canopy task with sparse rewards only. We also tested a direct PPO agent without any pre-trained module, similar to the one from our first baseline [3]: with a direct end-to-end approach, under-canopy fails without carefully shaped dense rewards, and its action histograms revealed unrealistic, hard to deploy behavior, confirming the benefits of the pre-trained controller. Once the best controllers were selected: a LPC and a state-dependent VQ-LPC; we tested the full pipeline. The new dataset combined six NMPC experts with the row-change heuristic. Controllers were re-trained successfully, with higher losses but valid structure. In Stage II, both reached up to 100% success rate on all three tasks, with the VQ-LPC training being slightly more stable. The OOD action rate remained around 1% for the LPC and below 0.7% for the VQ-LPC. Furthermore, in row-change task, the VQ-LPC was able to be trained with $w_a = 0$, therefore completing all the navigation tasks with sparse rewards only.

3.2. Sim-to-Real Transfer

The training environment in Isaac Lab implements the TerraSentia model dynamics on the PhysX engine. Faithful wheeled locomotion is notoriously hard in this framework, and many works drive wheeled platforms through a dedicated module rather than the physics engine. Those solutions bottleneck domain randomization; we therefore implement a sim-to-real pipeline to close the gap between the digital twin and the real robot instead of using an external module.
To close the locomotion gap, three corrections to the twin were applied:

- the wheel-actuator response time was reduced, by increasing joint damping, armature and friction, so the twin reaches its commanded velocity slightly faster than the real robot and the action commands do not oscillate;
- front and rear wheel speeds were equalized through per-joint motor tuning to remove lateral bias;
- the angular velocity command was cor-

rected by using RTK-GNSS corrected real robot's trajectories as ground-truth, since the twin completed maneuvers only at unrealistically high ω.

To close the perception gap, the readings from the 2D multi-channel LiDAR from simulation had to match approximately the real 3D LiDAR ones. With a statistical analysis of real scans, collected by teleoperating the robot in late-stage soy, field of view and channel count in simulation were fixed.

3.3. Field validation

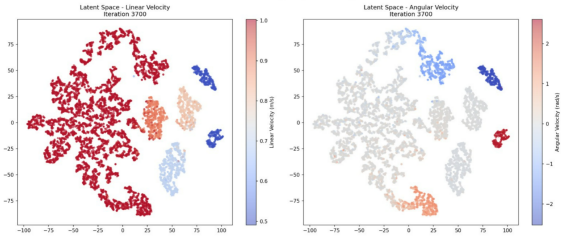

Figure 3: LPC latent space.

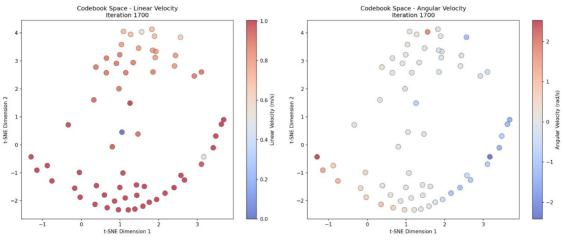

Figure 4: VQ-LPC codebook space.

Real expert demonstrations were collected across Campus 1 of EESC (Escola de Engenharia de São Carlos), São Carlos, SP, Brazil. We collected 30 NMPC demonstrations for a total of 16.95min, where each episode has minimum 13 waypoints reached, together with three episodes row change demonstration of 16.85min. All the episodes were collected on grassy terrain, and odometry was provided by Direct LiDAR-Inertial Odometry (DLIO), preferred over an RTK-GNSS pipeline because of heading oscillations. We then proceeded to train the two controllers validated in simulation, with the same network inputs and structure. The results of training are much better than the ones obtained in multi-expert configuration. This is probably due both to the lower variety of DR achievable in real life and to the dataset size. In Figure 3 and 4, the well structured latent space of the two models is shown.

To validate the waypoint following task, we compared our models with the best NMPC configuration we found. We decided to test the models in three different types of terrain: a grassy terrain (the same used for data collection), a loose granular terrain, and asphalt. The results are exposed in Table 1. The trained controllers bring some improvement over the baseline NMPC in the configuration used for data collection. Furthermore the LPC was tested against the critical NMPC configurations on grassy terrain: low waypoint tolerance and close waypoint spawning range. In the first case the LPC success rate was 90% against the 20% of the NMPC, while in the second case 90% against 50%.

To validate the under-canopy navigation task, we trained the three different models:

- **Phantom**, the LPC controller where the Stage II encoder is the fine-tuned Crow [1] perception module.
- **VQ-Phantom**, the VQ-LPC controller with the fine-tuned Crow [1] perception module.
- **LPC-Occupancy**, the LPC controller where the Stage II encoder is trained from scratch in simulation using the occupancy grid pipeline.

All those models are tested in a real agricultural scenario in Embrapa Instrumentação (São Carlos, Brazil), and compared with the project's baseline for under-canopy navigation: Crow. Because of the late season, the two agricultural tasks could be validated only on late-stage soy. The four aforementioned models are deployed on four different rows, where each row had six dedicated experiments. The average hits per row is exposed in Table 2. We can conclude that the pre-trained perception backbone improves significantly the under-canopy performances, with Phantom models performing far better than the LPC-Occupancy model. Also, Crow model performs overall worse than Phantom, meaning that the domain randomization and the avoidance of an interface in between controller and perception module strongly helps the model.

For the changing row task, the tests happened over 5 different exit rows, where the same row was picked twice per module. The tested mod-

Controller	Grass	Granular	Asphalt	Succ.
NMPC	9/10	8/10	9/10	86.67%
LPC	10/10	9/10	10/10	96.67%
VQ-LPC	9/10	9/10	10/10	93.33%

Table 1: Waypoint-following success across terrains.

ules were this time only ours, and the results are shown in Table 3. The Phantom modules completed up to 80% of the trials, whereas the one-shot occupancy-grid encoder, effective in simulation, never succeeded in real soy.

Model	R1	R2	R3	R4
Phantom	**0.67**	**1.83**	1.83	**1.00**
VQ-Phantom	1.00	2.67	**1.67**	1.83
Crow	2.67	3.50	2.17	1.33
LPC-Occupancy	2.67	4.33	3.50	3.17

Table 2: Under-canopy navigation model performance in late-stage soy.

Model	Task completions
Phantom	**80%**
VQ-Phantom	70%
LPC-Occupancy	0%

Table 3: Row-change completion rate.

4. Conclusions

This thesis introduced Phantom, a two-stage hierarchical learning process for wheeled robot navigation validated on the TerraSentia robot. Our idea was to first train a controller by shaping expert knowledge into a structured latent space using privileged information, and then to reuse that frozen module with exteroceptive information to solve several distinct navigation tasks. We demonstrated the following improvements against the baselines of this project:

- With respect to the **direct end-to-end** approach from [3], Phantom provides a broader DR suite, a validated sim-to-real pipeline, and the ability to generalize to sparse-reward navigation tasks.
- With respect to the **NMPC**, Phantom removes the dependence on an accurate state estimate, making it more robust to uncertainty and efficient across different terrains and configurations.
- With respect to **Crow**, Phantom avoids information loss and, thanks to domain randomization, is robust to poor LiDAR readings.

Future works may try to assemble a complete navigation system that performs both under-canopy and row-change, or to include past observations in the network pipeline.

References

[1] Francisco Affonso, Felipe Tommaselli, Gianluca Capezzuto, Mateus Valverde Gasparino, Girish Chowdhary, and Marcelo Becker. Crow: A self-supervised crop row navigation algorithm for agricultural fields. *Journal of Intelligent Robotic Systems*, 111, 02 2025.

[2] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019.

[3] Ana Luiza Mineiro, Francisco Affonso, and Marcelo Becker. End-to-end crop row navigation via lidar-based deep reinforcement learning, 2025.

[4] Aaron Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA*, 11 2017.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.