Bachelor's Thesis

Submitted in 2025

End to End Learning-Driven Navigation for Agricultural Robots

Author

Felipe Andrade Garcia Tommaselli

Department of Electrical and Computer Engineering - EESC/USP

Supervisor

Prof. Dr. Marcelo Becker

Department of Mechanical Engineering - EESC/USP

São Carlos, SP, Brazil

June 2025

Bachelor's Thesis

Submitted in 2025

End to End Learning-Driven Navigation for Agricultural Robots

Monograph submitted to the Electrical Engineering program, Electronics emphasis, at the São Carlos School of Engineering of the University of São Paulo, in partial fulfillment of the requirements for the degree of Electrical Engineer.

Advisor: Prof. Assoc. Marcelo Becker

São Carlos, SP, Brazil

June 2025

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Tommaselli, Felipe Andrade Garcia
T661e End to End Learning-Driven Navigation for Agricultural Robots / Felipe Andrade Garcia Tommaselli; orientador Marcelo Becker. São Carlos, 2025.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2025.

1. Agricultural Robotics. 2. Autonomous Systems. 3. Reinforcement Learning. 4. Model-Based Control. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Felipe Andrade Garcia Tommaselli

Título: "End to End Learning-Driven Navigation for Agricultural Robots"

Trabalho de Conclusão de Curso defendido e aprovado em 26 / 06 / 2024,

com NOTA 10,0 (DEZ,), pela Comissão Julgadora:

Prof. Associado Marcelo Becker - Orientador SEM/EESC/USP

Prof. Titular Marco Henrique Terra - SEL/EESC/USP

Dr. Leonardo Bonacini - EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Professor Associado José Carlos de Melo Vieira Júnior

Acknowledgments/Agradecimentos

Aos meus pais, Antonio e Giselle, que me deram raízes fortes e asas ainda maiores. Obrigado por me ensinarem a coragem de partir com a certeza de que sempre terei para onde voltar.

À minha irmã, Beatriz, pela nossa conexão única e genuína em cada passo dessa jornada.

À minha companheira de vida, Brendha, por abraçar plenamente cada fase ao meu lado, tornando o caminho ainda mais especial e significativo.

Ao Prof. Marcelo Becker, pela orientação, apoio e confiança em minha capacidade ao longo da minha trajetória como pesquisador.

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo fomento durante os anos de Iniciação Científica e Estágio no Exterior.

Aos meus companheiros de apartamento 211, pelas memoráveis experiências que impulsionaram o trabalho aqui apresentado.

Por fim, agradeço aos colegas da Graduação e do Grupo SEMEAR por compartilharem comigo esta caminhada.

Contents

List of Figures

Nomenclature

API Application Programming Interface

BC Behavior Cloning

CEM Cross–Entropy Method

DDPG Deep Deterministic Policy Gradient

DOF Degrees of Freedom

EKF Extended Kalman Filter

EMA Exponential Moving Average

GNSS Global Navigation Satellite System

GPS Global Positioning System

GPU Graphics Processing Unit

iLQR iterative Linear Quadratic Regulator

IMU Inertial Measurement Unit

IQR Interquartile Range

LiDAR Light Detection and Ranging

LQR Linear-Quadratic Regulator

MBRL Model-Based Reinforcement Learning

MDP Markov Decision Process

MPC Model Predictive Control

MPPI Model Predictive Path Integral

NMPC Nonlinear Model Predictive Control

ODE Open Dynamics Engine

PID Proportional-Integral-Derivative

RL Reinforcement Learning

ROS Robot Operating System

RTK Real-Time Kinematic

SAC Soft Actor-Critic

Sim2Real Simulation-to-Real Transfer

SimNorm Simplicial Normalization

TD Learning Temporal Difference Learning

TD-MPC Temporal Difference Model Predictive Control

TD-MPC2 Temporal Difference Model Predictive Control, version 2

TD3 Twin Delayed Deep Deterministic Policy Gradient algorithm

Abstract

TOMMASELLI, F. ***End to End Learning-Driven Navigation for Agricultural Robots***. 2025. 72 p. Monograph (Course Conclusion Paper) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2025.

Agricultural phenotyping at scale requires autonomous robots capable of navigating densely planted crop rows without GNSS, posing significant challenges for traditional control approaches. This thesis introduces an end-to-end learning-driven navigation framework for under-canopy agricultural robots that fundamentally reconceptualizes the navigation problem through model-based reinforcement learning. In contrast to conventional perception-plus-controller architectures that fragment the navigation pipeline into isolated modules, we implement a unified Temporal Difference Model Predictive Control (TD-MPC2) system that simultaneously learns world dynamics, reward prediction, and action selection within a shared latent representation. Leveraging latent-space world modeling for compact environmental representation, our approach facilitates efficient Model Predictive Path Integral control with minimal overhead. Our learning-driven system, implemented on the TerraSentia under-canopy robot and adapted for crop-following, significantly outperforms current model-free baselines in diverse simulated field conditions. It achieved a 33% increase in successful row completion and a 43% gain in mean distance traversed compared with CropfollowRL, while an oracle variant supplied with ground-truth states reached 100% completion at the maximum 57.5m, demonstrating that remaining errors stem primarily from LiDAR feature extraction rather than control. While the LiDAR-based variant shows greater variability in performance (120m peak, average 36.98 $\pm$ 26.95m) and has yet to outperform the perception-plus-controller CROW benchmark, these findings highlight a clear direction: improving LiDAR-derived row-edge segmentation should close the remaining gap, enabling the unified TD-MPC2 framework to match or surpass state-of-the-art deterministic pipelines.

Keywords: Agricultural Robotics, Field Phenotyping, Autonomous Systems, Reinforcement Learning, Model-Based Control

Resumo Extendido

TOMMASELLI, F. ***End to End Learning-Driven Navigation for Agricultural Robots***. 2025. 72 p. Monografia (Trabalho de Conclusão de Curso) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2025.

A crescente demanda por alimentos impulsiona a agricultura moderna a buscar soluções inovadoras para otimizar a produção e a eficiência. Nesse cenário, a fenotipagem agrícola em larga escala surge como uma ferramenta crucial, permitindo a coleta massiva de dados sobre as características das plantas diretamente no campo. No entanto, essa tarefa apresenta desafios significativos, especialmente no que tange à navegação autônoma de robôs em meio a plantações densas e sob a copa das culturas, onde sinais de GNSS são frequentemente indisponíveis ou pouco confiáveis. Abordagens tradicionais de controle, que geralmente separam os módulos de percepção e planejamento, têm se mostrado limitadas para lidar com a complexidade e a dinâmica desses ambientes.

Esta tese propõe uma mudança de paradigma ao introduzir um framework de navegação de ponta a ponta, impulsionado por aprendizado por reforço baseado em modelos, para robôs agrícolas que operam sob o dossel das plantações. Diferentemente das arquiteturas convencionais, que fragmentam o processo de navegação, nossa abordagem implementa um sistema unificado de Controle Preditivo Baseado em Modelo por Diferença Temporal (TD-MPC2). Esse sistema é capaz de aprender simultaneamente a dinâmica do ambiente, prever recompensas futuras e selecionar as ações ótimas, tudo dentro de uma representação latente compartilhada e compacta do mundo. Essa modelagem em espaço latente não apenas simplifica a representação do ambiente, mas também facilita um controle preditivo eficiente com mínima sobrecarga computacional.

O sistema foi implementado e adaptado para a tarefa de seguimento de fileiras de culturas utilizando o robô TerraSentia. Ensaios extensivos em simulação — contemplando cenários com milho e sorgo sob diferentes níveis de oclusão e terreno — revelaram que a variante guiada por LiDAR completou 33,3% dos percursos, atingiu distância média de 36,98±26,95m e, em seus melhores episódios, percorreu 120m contínuos, superando o CropfollowRL, que não concluiu nenhuma trajetória e percorreu apenas 32,7m em média. Em contrapartida, a variante "oráculo", alimentada com estados de solo verdade, alcançou 100% de conclusão e percorreu consistentemente os 57,5m máximos, evidenciando que o gargalo reside na extração de características do LiDAR e não no planejador TD-MPC2.

Embora ainda fique abaixo do benchmark CROW (100% de conclusão e 57,5m), nosso método apresentou um ganho relativo de 13% na distância média e um aumento de 33 p.p. na taxa de conclusão sobre o melhor modelo-livre existente, resultados que apontam para um caminho claro de melhoria ao aprimorar a segmentação de bordas de fileira obtida a partir do LiDAR.

A robustez do sistema frente a variações ambientais, como mudanças na iluminação, oclusão de sensores por folhagem e irregularidades no terreno, é uma das principais contribuições deste trabalho. Ao unificar percepção e controle em um único framework de aprendizado, o robô desenvolve a capacidade de se adaptar autonomamente a novas condições de campo com necessidade mínima de retreinamento. Isso é possível porque o agente aprende a tomar decisões baseadas em sua própria experiência e na previsão das consequências de suas ações, eliminando a necessidade de ajustes manuais complexos e heurísticas pré-definidas que caracterizam muitos sistemas atuais.

A metodologia desenvolvida não apenas avança o estado da arte em automação agrícola, mas também oferece um caminho promissor para superar o desafio da lacuna entre simulação e realidade (Sim2Real), um obstáculo persistente no desenvolvimento de robôs para ambientes complexos e não estruturados. A capacidade do sistema TD-MPC2 de aprender um modelo do mundo e planejar ações dentro desse modelo aprendido, mesmo com dados sensoriais brutos como entrada (LiDAR neste estudo), sugere uma maior generalização e adaptabilidade. A pesquisa foca em como um sistema pode perceber o ambiente, decidir o que fazer e agir de forma integrada e adaptativa, evitando as transições problemáticas entre diferentes modos de operação que limitam sistemas agrícolas mais antigos.

Em suma, este trabalho estabelece as fundações para uma nova geração de robôs agrícolas verdadeiramente autônomos, capazes de operar de forma eficiente e segura em condições de campo desafiadoras. Ao reconceitualizar o problema da navegação através de um prisma de aprendizado por reforço baseado em modelos e controle preditivo, demonstramos um salto qualitativo em termos de desempenho, robustez e adaptabilidade. As contribuições aqui apresentadas abrem novas perspectivas para a fenotipagem de precisão e para a automação inteligente no setor agrícola, com o potencial de transformar a maneira como interagimos e gerenciamos os agroecossistemas.

Keywords: Robótica Agrícola, Aprendizado por Reforço, Controle Baseado em Modelos, TD-MPC, Sistemas Autônomos, Fenotipagem de Campo

1 Introduction

Agricultural production, and maize cultivation in particular, is approaching a data-driven inflection point where genotype-to-phenotype mapping and precise crop management demand orders of magnitude more in-field measurements than human crews can realistically gather. Therefore, ground-based, under-canopy phenotyping platforms are now indispensable for enabling non-destructive, high-throughput trait collection directly within crop rows [3]. Yet the promise of these machines remains partially unrealized: narrow corridors, occluding foliage, and the critical need for sub-centimeter positioning accuracy for reliable navigation makes most Real-Time Kinematic (RTK) or Global Navigation Satellite System (GNSS) solutions ineffective. This forces researchers to confront the problem of autonomous locomotion head-on in areas without Global Positioning System (GPS) coverage, specially.

These agricultural platforms have evolved from manually pushed or driven systems to self-propelled, autonomous robots capable of navigating between crop rows with minimal human intervention. The transition to motorized and autonomous ground robots offers unprecedented flexibility and enables parallel operation of multiple units, dramatically increasing throughput for phenotypic data collection. Such autonomous systems can measure various traits in crops such as maize and sorghum, including plant height, volume, stem diameter, stem strength, and leaf area index, thus addressing critical bottlenecks in agricultural phenotyping [3].

Early systems addressed the navigation bottlenecks by tightly coupling dedicated perception pipelines, typically Light Detection and Ranging (LiDAR) scan matching or monocular row-geometry extraction, to classical controllers such as Proportional-Integral-Derivative (PID), Linear-Quadratic Regulator (LQR), or Model Predictive Control (MPC). Terrasentia, a compact four-wheel-drive robot designed at the University of Illinois, exemplifies this lineage and has become a benchmark chassis for under-canopy research [4]. Successive iterations improved robustness through sensor fusion and path-integrated odometry, but the underlying architecture remained bifurcated: perception generates an precise row centroid estimate, and a handcrafted controller tracks that curve.

Despite demonstrable field successes, perception-plus-controller stacks reveal structural weaknesses under real scenarios. Seasonal lighting changes, cultivar-specific canopy architectures and accumulated foliage obscure the row edges, causing LiDAR-based methods to fail intermittently [5]. Camera-centric pipelines also suffer from heavy occlusion or

low-texture stems, even when augmented with semantic keypoints [6]. Controller performance is equally dubious, gains tuned for a mid-season canopy often oscillate when soil firmness or wheel slip changes, compelling operators to re-calibrate frequently.

The research community has responded with more sophisticated perception modules: CROW leverages self-supervised visual neural networks to learn crop-row priors [5], while robust sensor-fusion frameworks blend stereo, Inertial Measurement Unit (IMU) and LiDAR cues to achieve decimeter level accuracy [7]. Nevertheless, these advances also treat perception and control as separable concerns. Their integration still relies on deterministic feedback laws that do not learn from navigation experience, making zero-shot deployment to unseen fields hard to find.

Reinforcement Learning (RL) reshapes this landscape by casting navigation as sequential decision-making under uncertainty. Model-free RL has already demonstrated the feasibility of end-to-end policy optimization on Terrasentia in CropFollowRL [8], where a convolutional encoder maps front-facing images directly to steering commands. While encouraging, purely model-free approaches incur prohibitive sample complexity and tend to over-fit to the training canopy structure, necessitating extensive real-world data collection or risky sim-to-real transfer.

RL agents learn to map observations to actions by interacting with their environment and receiving scalar reward signals, eliminating the need for explicit programming of control laws or detailed system models. This data-driven approach holds significant promise for agricultural robotics, offering the potential to develop more adaptive and generalized navigation policies. Unlike traditional methods that separate perception and control, RL can learn end-to-end policies that directly map raw sensor inputs to control outputs, potentially leading to more reactive and robust navigation behaviors.

Model-based reinforcement learning (MBRL), especially methods embedding Model Predictive Control (MPC), offers a compelling alternative. By learning a latent dynamics model and leveraging on-line planning, such algorithms inherit MPC's look-ahead reasoning while continuously refining the world model from new experience. Temporal-Difference Model Predictive Control (TD-MPC) and its successor TD-MPC2 constitute the state-of-the-art in this paradigm [9, 2]. TD-MPC2 constructs a control-centric latent space, rolls out thousands of candidate actions with an Model Predictive Path Integral (MPPI) optimizer, and updates its value estimates via temporal-difference learning (TD Learning), yielding a single self-tuning policy/planner capable of rapid adaptation across tasks.

Our implementation adapts the TD-MPC2 framework to the specific challenges of under-canopy agricultural navigation. By utilizing only egocentric sensor data, such as LiDAR readings and potentially images from the robot's onboard camera, our system learns to navigate solely from the robot's perspective. This is a significant departure from many traditional methods that rely on global localization or pre-built maps, adding a layer of complexity that our end-to-end learning approach is designed to handle. The system learns a policy that directly maps raw sensor inputs to robot control commands (e.g., linear and angular velocity). The adaptation of TD-MPC2 involved careful consideration of the observation space, action space, and reward function to effectively capture the nuances of navigating within crop rows while avoiding collisions and maintaining a desired trajectory.

The main goal of this line of research is to create a clear path for future work on model-based systems that intend to leverage zero/few-shot learning on deployment field. For this, we introduce this worldwide validated landscape to agricultural robots on a detailed framework. The Simulation-to-Real Transfer (sim2real) gap, as illustrated in figure 1, remains unsolved, however, most of the methods discussed in this article aim to mitigate it.

Figure 1: The figure illustrates the Sim2Real transition of the TerraSentia robot. The left shows the TerraSentia in a Gazebo simulation environment [1], while the right depicts the real robot performing the same task in a plantation.

A well-known challenge in Reinforcement Learning is its sensitivity to parameter tuning, which often varies significantly across tasks and domains [10, 11, 12, 9]. This system design prioritizes components that enhance stability in unseen environments, requiring minimal to no parameter adjustments. This approach aligns with key insights from foundational works, particularly [2] and [13].

Additionally, to improve zero-shot capabilities across diverse scenarios, addressing Out

of Distribution (OOD) actions is crucial. OOD actions refer to input-output mappings poorly represented during training, which can lead to deployment failures. To mitigate this, we implement carefully selected techniques, such as Behavior Cloning (BC), to enhance stability and ensure reliable zero-shot deployment.

Preliminary results indicate that the LiDAR-based variant of TD-MPC2 completed 33.3% of the trajectories, covered an average of 36.98 ± 26.95 m, and reached 120 m in the best episode—outperforming the CropfollowRL baseline, which completed no trajectories and reached only 32.7 m on average, a relative gain of roughly 13% in distance traveled [8]. In addition, the oracle variant fed with ground-truth states achieved 100% completion and consistently covered the maximum 57.5m, showing that the remaining bottleneck lies in LiDAR feature extraction rather than in the TD-MPC2 planner.

Despite this progress, the deterministic CROW pipeline, which also achieves 100% completion and 57.5m, remains the benchmark to surpass [5]. The high variability of our LiDAR approach (standard deviation of 26.95 m) indicates a need to improve row-edge segmentation. Once this limitation is mitigated, the unified TD-MPC2 framework is expected to match or exceed CROW's performance while preserving the benefits of online adaptation and reduced dependence on manual calibration.

In general, we acknowledge that our approach yields less compelling results than Crop-follow++ or CROW [6, 14]. This work serves as a foundational step within the RL-based framework category, aiming to advance research in this field beyond heuristic baselines. Work should be understood as a deeper step into the RL-based framework category, with the aim of further increasing work in this field until the heuristic baseline barrier is passed.

These findings support a broader thesis: controller theory need not be abandoned but absorbed into learning systems. By embedding MPC inside the RL agent, we preserve the long-horizon optimality guarantees of planning while discarding brittle, manually tuned control loops. The resulting architecture scales consistently with sensor modalities and field variability, charting a path toward few-shot or even zero-shot deployment in novel fields.

This thesis makes several significant contributions to the field of agricultural robotics and autonomous navigation. We highlight some key considerations:

- We present the first end-to-end TD-MPC2 implementation on the Terrasentia platform, replacing separate perception and control modules with a single world-model planner-policy that learns directly from raw onboard sensors.

- The demonstration that effective low parameter tuning is feasible, where the orchestration of a really stable and well chosen system can be trained and deployed with little to no empirical parameter tuning.
- We establish a comprehensive discussion about uncertainty handling task-oriented for the crop follow, where we managed to incorporate multiple methods to enhance stability even on OOD actions.

2 Related Work

Recent advancements exemplified by CropFollow++ [6] have established robust frameworks for vision-based navigation through semantic keypoint extraction, achieving 767 meters between human interventions across 25 km deployments in cover crop planting operations. This foundational work, complemented by high-throughput phenotyping systems [3] that have systematically collected morphological data from nearly 200,000 experimental units across 142 fields, represents the current state-of-the-art benchmark for autonomous agricultural locomotion, progressing from 27 meters to 3629 meters between interventions through iterative system refinement.

Early attempts to transcend ad-hoc visual pipelines adopted self-supervision or geometric priors. *CROW* [5] employs LiDAR line-fitting to bootstrap its own waypoint labels, showing resilience to illumination and crop morphology, while *Navigating with Finesse* couples ResNet feature extraction with Ierative Linear Quadratic Regulator (iLQR) to optimize smooth trajectories through occlusions [14]. Both frameworks still rely on per-row geometric reasoning and hand-tuned recovery behaviors, highlighting the gap between classical control guarantees and practical stochastic field conditions.

Other precedents further illustrate this dichotomy. The heuristic filtering strategy of *Under-Canopy LiDAR Navigation* [4] mitigates outliers from foliage but struggles when inter-row clutter becomes indistinguishable from crops. Vision-only steering in *CropFollow* [15] offers a low-cost alternative. Yet its end-to-end convolutional policy cannot explicitly reason about future constraints, producing oscillatory behavior under dense occlusions. These systems have been augmented through sophisticated multi-sensor fusion techniques [7] that integrate Sensors and IMU via Extended Kalman Filter (EKF) frameworks, achieving navigation without interventions for distances up to 386.9 meters in contiguous field environments through probabilistic state estimation and uncertainty propagation methodologies.

Reinforcement learning has emerged as a transformative paradigm in autonomous navigation, with seminal work [16] demonstrating rapid adaptation of lane-following behaviors from randomly initialized parameters using monocular imagery within a single training day. This approach has been extended through goal-driven exploration systems [17] that combine deep reinforcement learning for local navigation with global waypoint selection algorithms, enabling autonomous environmental mapping without a priori knowledge representations. Modular architectures for driving policy transfer [18] have further

established methodological frameworks for bridging the simulation-reality gap through abstraction layers that insulate learned policies from direct exposure to sensor modalities and physical dynamics, facilitating cross-platform deployment without extensive parameter recalibration.

The CropFollowRL framework [8] represents a significant advancement in applying reinforcement learning to agricultural navigation challenges through a real2sim2real methodology incorporating semantic keypoints abstraction. This implementation employs the Twin Delayed Deep Deterministic Policy Gradient algorithm (TD3) trained in Gazebo environments [1], with a perception architecture comprising fully convolutional networks predicting triangular vertices from crop rows represented as a 13442-dimensional state vector. The reward formulation incorporates significant negative reinforcement for collision events (-100) balanced with positive reward signals derived from forward velocity metrics, demonstrating successful deployment across multiple field robots while establishing baseline performance metrics ranging from 9.07m to 30m of intervention-free navigation in diverse simulated agricultural environments.

The limitations of model-free approaches in sample efficiency and generalization capabilities motivated our investigation into Model-Based frameworks. Our chosen architecture, a MBRL with a planner, is a well-established approach, notably demonstrated by PETS which was proposed in 2018 and is widely recognized in the quadruped robotics community [19, 20]. This research direction remains active and continues to evolve and optimize; in that sense, the current in development and state-of-the-art TD-MPC architecture was chosen as backbone for this work.

The original TD-MPC framework [9] synergistically combines model-free and model-based reinforcement learning paradigms through joint learning of task-oriented latent dynamics models and terminal value functions via temporal difference learning. This hybrid approach performs trajectory optimization over finite horizons using learned dynamics while leveraging terminal value functions to estimate long-term returns beyond the planning horizon. TD-MPC2 [2] represents an architectural evolution incorporating "decoder-free" latent space operations [21], LayerNorm stabilization, SimNorm regularization, and multiple Q-function ensembles, while employing maximum-entropy reinforcement learning as a policy prior for improved exploration across state-action manifolds. The main TD-MPC adaptations suitable for the Cropfollow task include high adaptability with minimal parameter tuning, as extensively demonstrated in TD-MPC2 in the multi-task benchmark [2] and in FOWM for few-shot real-world scenarios [10].

The model-based reinforcement learning landscape encompasses alternative methodologies that offer complementary insights, including unsupervised reinforcement learning from pixel representations [22] employing dual-phase training with unsupervised pre-training followed by task-specific fine-tuning. Physics-informed approaches such as Dyna-PINN [23] integrate domain-specific knowledge through physics-informed neural networks combined with Deep Dyna-Q reinforcement learning, encoding governing equations and physical constraints to enhance model generalization in data-sparse regimes.

More recently, DWL [24] and RWM-O [25], both represent different approaches, but all incorporate some sort of known techniques to enhance sim2real transition. Both algorithms represent state-of-the-art works for humanoids and quadrupeds locomotion, reinforcing the thesis that MBRL is becoming more relevant each year across different fields. It is worth noticing RL2AC [26], which proposes a hybrid approach with a controller, but this time with adaptive control. This article focuses heavily on rapid adaptation to unseen environments and efficiently addresses some concerns shared in the work presented here.

Our methodological approach draws inspiration from advances in FOWM [10], addressing distribution shifts between simulation environments and physical deployments (or even on different simulation environments) through epistemic uncertainty quantification and behavioral regularization. By employing a Q-ensemble of value functions to characterize model confidence across the state space, we constrain exploration to regions where the model exhibits high certainty, thereby enhancing operational safety and performance in complex agricultural environments characterized by dense vegetation, irregular terrain features, and dynamic illumination conditions. This uncertainty-aware approach has demonstrated particular efficacy in scenarios where environmental variations exceed those encountered during training phases, enabling robust generalization across diverse field morphologies through principled uncertainty-guided exploration.

The final component of our methodology incorporates policy constraint mechanisms inspired by TD-M(PC)^2 [13], which addresses value overestimation phenomena through Kullback-Leibler divergence minimization between learned and behavior policies. Our approach combines Behavior Cloning (BC) with an uncertainty quantification framework to reduce testing of out-of-distribution actions. Behavior cloning encourages the policy to generate actions similar to those in the populated buffer, while the uncertainty framework further constrains behavior. This dual mechanism is effective given our task's finite and well-defined action space, where extreme movements are rarely needed, promoting a consistent and reliable pace.

Synthesising the above discourse, our thesis positions the Terrasentia navigation challenge at the confluence of sensor-based agricultural robotics and sample-efficient model-based RL. By adopting a TD-MPC2 backbone augmented with behaviour cloning and uncertainty gating, We aim to combine how a system sees (perception), decides what to do (planning), and acts (control) into one adaptable method. This avoids the awkward shifts between different modes that limit older agricultural systems and moves us closer to completely automated operation under plant canopies. While many related studies influenced design decisions, this section highlights the most contributing works.

3 System Design

To enhance understanding of the project's system design, this section is divided into the following: Inference-Time Architecture, Algorithmic Pipeline Overview, Simulation Environment, and Training-Time Methods. The Inference-Time Architecture primarily focuses on the final deployed and runtime system. This structure first provides a clear overview of the end product, followed by detailed explanations of the algorithmic components, the simulation environment. Finally, we present the learning approach, highlighting its key aspects and differences from the inference-time architecture.

3.1 Inference-Time Architecture

The primary objective of this line of research is to develop a deployment-ready agricultural navigation system capable of few-shot adaptation in previously unseen environments. Here, the presented system aims to show a clear path towards this goal. This section presents a high-level overview of our system's inference-time architecture.

During inference, the Terrasentia robot is expected to enter previously unseen crop rows, operate without external positioning (e.g., GPS), self-adapt from the first control cycle onward. To accelerate the deployment process safely in unseen simulation environments, or even in real-world scenario, we can deploy the world model and policy already optimized in simulation together with a replay buffer pre-populated with simulation trajectories. This design can significantly reduce the traditional sim2real gap [27, 16], enabling the agent to learn critical navigation parameters on the very first stages of deployment.

The navigation task, hereafter *cropfollow*, is formalized as a corridor-traversal problem in which the robot must maximize the distance traveled while avoiding collision with row dividers. In typical maize or sorghum plots the corridor width falls below the robot's wheel track, producing frequent foliage occlusions and strong specular reflections that hinder sensor-only perception [3, 15]. Previous work tackles cropfollow through modular pipelines that estimate the corridor centreline from cameras or LiDAR and hand the estimate to an MPC or iLQR controller [6, 14, 7]. Our design discards this decomposition and learns a single world-model-powered policy, reducing explicit model-based control engineering while retaining long-horizon planning capability.

As depicted in Figure 2, the deployed system contains three tightly coupled components: (i) a latent-space world model f_θ learned with TD-MPC2 [2]; (ii) a sampling-based

planner, Model-Predictive Path Integral (MPPI), which rolls out action sequences inside f_θ; and (iii) a stochastic policy prior π_θ that biases the MPPI sampler and supplies the terminal-value estimate for each rollout. The only external input is a 2-D LiDAR scan (1081 points).

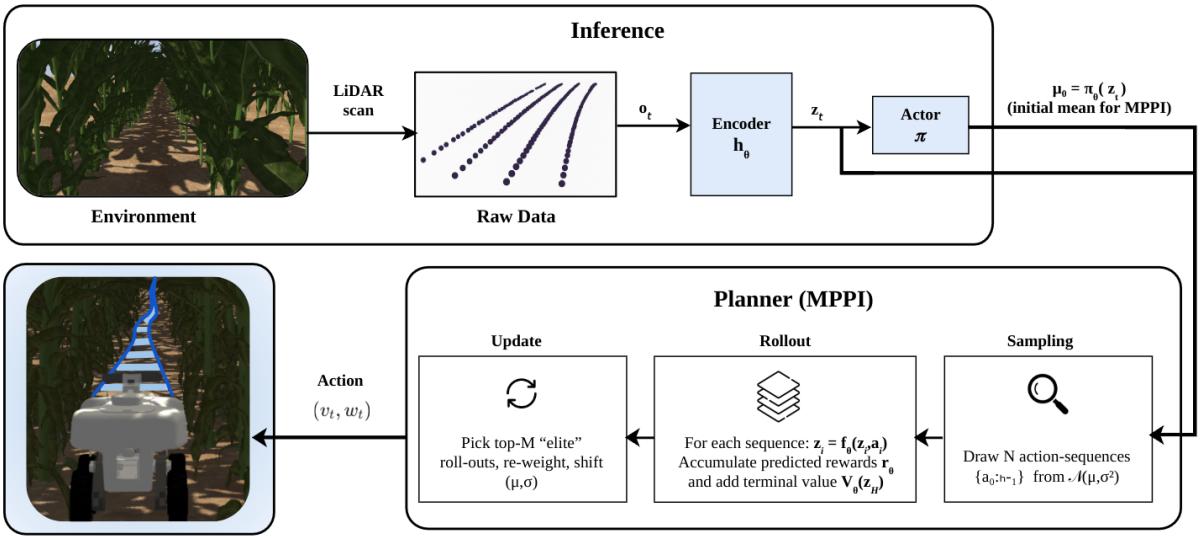

Figure 2: A single 1081-point LiDAR scan is distilled by the encoder h_θ into a latent z_t, which is the **only** state on which the rest of the stack operates. The stochastic actor π_θ is queried once per control cycle to provide the initial Gaussian mean and a handful of "π-trajectories," seeding the Model-Predictive Path-Integral (MPPI) planner with prior knowledge while preserving exploration entropy. MPPI then samples hundreds of additional sequences, rolls every candidate through the learned world-model trio $\{f_\theta, r_\theta, Q_\theta\}$ to obtain predicted rewards and a boot-strapped terminal value, and iteratively shifts its distribution toward the highest-return (elite) roll-outs. After a few inner iterations the planner outputs the best horizon-H sequence, and only its first control $\mathbf{a}_0$ is executed on the robot, while the remainder, together with the refined Gaussian statistics, warm-start the next cycle.

where $\mathbf{o}_t$ is the normalized scan and $\mathbf{z}_t$ the latent belief state. This contrasts with prior art that first estimates geometric state before invoking control [4]. Because the same latent dynamics underpin both planning and policy evaluation, no explicit state estimator is required. At each control cycle the current observation is embedded into a latent state $z_{t^*} = h(s_t)$, giving a fresh posterior that supersedes the model's one-step prediction. The agent then performs local trajectory optimization in this latent space using MPPI, specifically, it samples a population of action sequences $a_{t:t+H-1}$ from a time-dependent multivariate Gaussian distribution, with a portion of these sequences biased by a learned

policy prior π_θ (the actor).

We frame the crop-following task as a Markov Decision Process, a standard framework for sequential decision-making. Our system's situation at any given moment is represented by a latent state, encompassing relevant information about the environment and the robot's configuration. The robot's actions are continuous, consisting of a desired forward velocity and an angular velocity. We operate without a perfect, explicit model of how these actions affect the state, instead relying on an unknown transition mechanism. The robot is driven through a reward function that encourages movement along the crop rows while discouraging deviations from the desired path and preventing collisions. The ultimate goal is to maximize the expected sum of future rewards, discounted to prioritize immediate gains, reflecting a preference for near-term performance.

We leverage the learned function, denoted by f_θ, not just for taking actions but also as a predictive model of system dynamics. This allows us to anticipate the consequences of our actions. This integration is key: the parameters of our predictive model and our decision-making policy are learned jointly.

Although experiments in this thesis employ LiDAR exclusively, the architecture is sensor-agnostic: a camera front-end may replace or supplement the range input with minor encoder modifications, and preliminary simulation indicates that additional pose cues further improve sample efficiency. All results here are obtained in high-fidelity Gazebo worlds that replicate maize and sorghum geometries [1]. Nonetheless, the inference pipeline presented above is already compatible with the real Terrasentia's onboard Jetson Orin and requires minimum to no modifications before field trials.

3.2 Algorithmic Pipeline Overview

The algorithmic foundation of our agricultural robot navigation system is built upon the principles of model-based reinforcement learning (RL), specifically the Temporal Difference Model Predictive Control (TD-MPC) framework. Our system adapts TD-MPC2 [2] for under-canopy navigation. The system architecture consists of the following core components:

- **World Model**: A neural network-based implicit model that learns environment dynamics, predictive rewards, and value estimates without explicitly reconstructing observations.

- **Encoder**: Transforms high-dimensional sensory inputs (e.g., camera images, LiDAR scans) into a compact latent representation.
- **Latent Dynamics**: Predicts the next latent state given the current state and action.
- **Reward Predictor**: Estimates the immediate reward associated with state-action pairs.
- **Value Function**: Approximates the expected long-term return from a given state.
- **Policy Network**: Provides action priors to guide the planning process.
- **Model Predictive Control Planner**: Optimizes action sequences using the learned world model.

These components operate in concert, forming a closed-loop system where the robot continuously perceives its environment, updates its internal model, plans optimal trajectories, and executes actions.

3.2.1 Foundations: Model-Based Reinforcement Learning and TD Learning

We formalise the navigation task as a continuous-state, continuous-action Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R, \gamma)$ with discount factor $\gamma \in (0, 1)$. At each discrete time step t, the agent observes $s_t \in \mathcal{S}$, selects $a_t \in \mathcal{A}$, transitions according to $s_{t+1} \sim T(\cdot \mid s_t, a_t)$, and receives reward $r_t = R(s_t, a_t)$.

Model-based reinforcement learning proceeds by fitting a parametric dynamics model $\hat{T}$ of the true transition kernel T and using it for look-ahead planning. Temporal-difference (TD) learning, on the other hand, updates the action-value estimator Q_ϕ by minimising the one-step TD error. Define the TD error

$$\delta_t \;=\; r_t + \gamma \max_{a'} Q_{\bar{\phi}}(s_{t+1}, a') \;-\; Q_\phi(s_t, a_t),$$

and the squared-error loss $L_t = \frac{1}{2}\,\delta_t^2$. A stochastic-gradient step on this loss gives

$$\phi \;\leftarrow\; \phi \;+\; \eta\,\delta_t\,\nabla_\phi Q_\phi(s_t, a_t), \tag{3.1}$$

where $\bar{\phi}$ denotes a slowly updated target network and η is the learning rate.

TD-MPC couples these two paradigms by rolling out imagined trajectories inside the learned model and then bootstrapping beyond the MPC horizon with Q_ϕ. This hybrid

objective reduces long-horizon bias and yields strong data efficiency on high-dimensional continuous-control tasks [9].

Our implementation inherits the essential mathematical structure but introduces domain-specific modifications described in Section 3.2.7.

3.2.2 The TD–MPC2 Architecture

The latent world model of TD-MPC2 is factorized into five neural modules:

- **Encoder** $h_\theta : \mathcal{S} \to \mathcal{Z}$ projects raw observations into a compact latent state.
- **Dynamics** $d_\theta : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$ predicts the next latent state.
- **Reward head** $R_\theta : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}^B$ outputs a vector of logits over a *logarithmically spaced* support $\mathcal{R} = \{ r^{(1)}, \ldots, r^{(B)} \}$. A scalar reward $r \in \mathbb{R}$ is encoded as a *two-hot* target: probability mass is placed on the two neighbouring bins $(r^{(j)}, r^{(j+1)})$ that bracket r in proportion to their distance from r. Training therefore minimises a soft cross-entropy loss, while inference recovers the expected reward $\hat{r} = \sum_j \mathrm{softmax}(R_\theta)_j \, r^{(j)}$. The same discretised representation is used for the Q-values.
- **Value ensemble** $\{ Q_\theta^k \}_{k=1}^5$ predicts terminal returns; the TD target is the minimum of two randomly chosen heads to reduce over-estimation bias.
- **Policy prior** $\pi_\theta : \mathcal{Z} \to \mathcal{A}$ supplies a proposal distribution for MPPI and accelerates its convergence.

Figure 3 from [2] illustrates the essential role of latent space transformation in predictions and training, as it is used for all estimations.

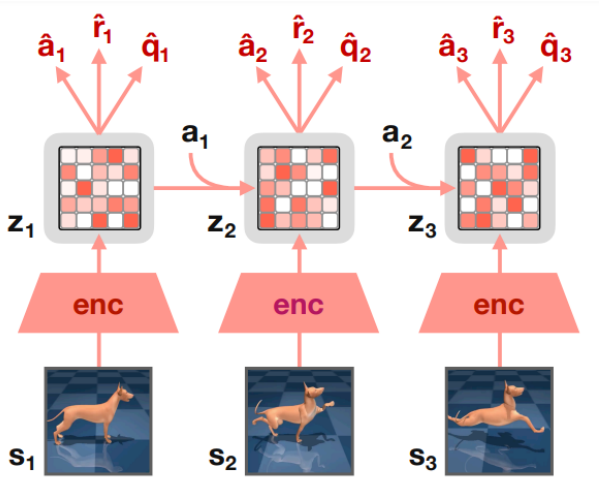

Figure 3: The TD-MPC2 architecture ([2]) encodes observations (s) into a latent space (z). Within this latent space, the model predicts future actions ($\hat{a}$), rewards ($\hat{r}$), and terminal values ($\hat{q}$), eliminating the need to decode future observations. All core operations are performed exclusively in the latent domain.

All sub-networks are multilayer perceptrons with LayerNorm and Mish activations. Layer activations are projected onto a probability simplex using a softmax function: $z' = \text{softmax}(W_{proj} \cdot z) \cdot C$ using the Simplicial Normalization *SimNorm* operator to mitigate gradient explosion and encourage sparsity.

Training minimises a composite loss

$$\mathcal{L} = \sum_{t=0}^{H} \lambda^t \Big(\underbrace{\|z'_t - \text{sg}(h(s_{t+1}))\|_2^2}_{\text{joint-embedding}} + \text{CE}(R_\theta(z_t, a_t), r_t) + \text{CE}(Q_\theta(z_t, a_t), q_t) \Big), \tag{3.2}$$

with TD target $q_t = r_t + \gamma Q_{\bar{\theta}}\big(d_\theta(z_t, a_t), p_\theta(d_\theta(z_t, a_t))\big)$.

The policy prior is optimised with a soft-actor–critic objective

$$\mathcal{J}_\pi = \mathbb{E}_{z_t \sim \mathcal{B}}\Big[Q_\theta\big(z_t, \pi_\theta(z_t)\big) + \beta\, \mathcal{H}\big(\pi_\theta(\cdot \mid z_t)\big) \Big],$$

where the temperature β is tuned automatically, and Q_θ is rescaled online (RunningScale) to keep its magnitude comparable to the entropy bonus. A behaviour-cloning prior adds an ℓ_2 penalty $\lambda_{\text{BC}} \| a_{\text{policy}} - a_{\text{data}} \|_2^2$ to $\mathcal{J}_\pi$ [28, 2].

Compared with TD-MPC, version 2 increases model capacity by two orders of magnitude and employs multi-task embeddings to share statistical strength across domains [2].

3.2.3 Model Predictive Control Integration

Model Predictive Control (MPC) is an advanced, optimization-based control framework that explicitly incorporates a system's dynamics, performance objectives, and hard constraints into a single finite-horizon planning problem. At each control step, MPC solves an open-loop optimal control problem over a horizon of length H, generates a sequence of candidate actions, and applies only the first action before re-planning at the next step [29].

At decision time t we maintain a nominal control sequence $\mathbf{u}_{t:t+H-1} = \{u_t, \dots, u_{t+H-1}\}$. MPPI improves this sequence by drawing N trajectory roll-outs with exploratory noise $a_{t+k}^{(i)} = u_{t+k} + \sqrt{\Sigma}\,\varepsilon_k^{(i)}$, $\varepsilon_k^{(i)} \sim \mathcal{N}(0, I)$, and by re-weighting them according to their (discounted) return. The optimization objective that each iteration *pushes toward* is

$$\arg\max_{\mathbf{u}_{t:t+H-1}} \mathbb{E}_{\varepsilon_{0:H-1}}\Big[\sum_{k=0}^{H-1} \gamma^k\, R\big(z_{t+k}, a_{t+k}\big) + \gamma^H\, Q\big(z_{t+H}, a_{t+H}\big)\Big], \tag{3.3}$$

subject to the (known) dynamics $z_{t+k+1} = T\big(z_{t+k}, a_{t+k}\big)$, with $z_t = s_t$ the measured state. The terminal action is *not* part of the decision vector; instead, it is sampled from the current feedback policy,

$$a_{t+H} \;\sim\; p_\theta\big(\,\cdot \mid z_{t+H}\big),$$

so that $Q(z_{t+H}, a_{t+H})$ represents the soft value of following p_θ beyond the finite horizon.

After evaluating the returns $S^{(i)} = \sum_{k=0}^{H-1} \gamma^k R(z_{t+k}^{(i)}, a_{t+k}^{(i)}) + \gamma^H Q(z_{t+H}^{(i)}, a_{t+H}^{(i)})$, the nominal controls are updated according to the path-integral weight $w^{(i)} \propto \exp\big(\frac{1}{\lambda} S^{(i)}\big)$:

$$u_{t+k} \;\leftarrow\; u_{t+k} + \sum_{i=1}^{N} w^{(i)} \sqrt{\Sigma}\,\varepsilon_k^{(i)}, \quad k = 0, \dots, H-1.$$

Only the first control u_t is applied to the real system; the horizon then slides forward and the procedure repeats at $t+1$.

In TD-MPC2 the dynamics T is replaced by a learned model d_θ, the terminal value V by a learned action-value Q_θ, and optimization is performed with a Model Predictive Path Integral (MPPI) sampler that iteratively refines a diagonal Gaussian

$$\mu \leftarrow \mu + \eta_\mu \frac{\sum_{i=1}^N w_i(\mathbf{a}_i - \mu)}{\sum_{i=1}^N w_i}, \qquad \sigma \leftarrow \sigma + \eta_\sigma \frac{\sum_{i=1}^N w_i\big((\mathbf{a}_i - \mu)^2 - \sigma^2\big)}{\sum_{i=1}^N w_i}, \tag{3.4}$$

where weights $w_i \propto \exp\big((\mathcal{R}_i - \max_j \mathcal{R}_j)/\tau\big)$ depend on rollout returns $\mathcal{R}_i$ and temperature τ.

The first control in the elite sequence is executed while the remainder serves as a warm-start for the next cycle, thereby ensuring real-time feasibility and closed-loop robustness.

To solve the MPC optimization problem efficiently, Model Predictive Path Integral (MPPI) control [30] was implemented, a derivative-free sampling-based method. MPPI iteratively:

- Samples multiple action sequences from a proposal distribution (initially guided by the policy prior)
- Evaluates each sequence by simulating trajectories through the learned world model
- Updates the proposal distribution based on the evaluated returns
- Repeats until convergence or a computational budget is exhausted

Mathematically, MPPI estimates parameters μ, σ of a time-dependent multivariate Gaussian with diagonal covariance:

$$\mu, \sigma = \arg\max_{(\mu,\sigma)} \mathbb{E}(a_t, at+1, ..., a_{t+H}) \sim \mathcal{N}(\mu, \sigma^2) \left[\gamma^H Q(z_{t+H}, a_{t+H}) + \sum_{h=t}^{H-1} \gamma^h R(z_h, a_h) \right] \tag{3.5}$$

On the Terrasentia chassis, stable steering in dense maize rows was achieved with an MPC horizon greater than $H = 12$, consistent with average plant spacing reported in [3]. However, for the TDMPC architecture during training longer horizons were computationally expensive and offered no planning improvement over shorter ones due to frequent recomputation. Following TDMPC2 guidelines and empirical results [2], the optimal performance was observed with a seemingly short horizon of $H = 3$. Closer examination reveals this horizon is sufficient for generating effective control actions efficiently.

3.2.4 Latent-Space World Modelling

A foundational element of our TD-MPC2 implementation is the use of latent space representations, which provide a crucial abstraction layer between diverse sensory inputs and the navigation algorithm. This approach enables sensor agnosticism, allowing our system to operate with various input modalities such as cameras, LiDAR, or combinations thereof. The encoder network h transforms high-dimensional observations s into a compact latent representation $z = h(s)$. For agricultural robots like TerraSentia, observations can include: Camera images, LiDAR scans or even Proprioceptive data such as wheel encoder readings and IMU measurements.

Instead of separate algorithms for each sensor, a latent space learns a unified, navigationally relevant representation, useful in agriculture where sensor reliability varies with

conditions (cameras in low light/occlusion, LiDAR with dust/foliage). The latent space representation offers several key advantages [21]:

- **Dimensionality Reduction**: High-dimensional inputs (e.g., images with thousands of pixels) are compressed into a compact latent vector (typically 128-512 dimensions), reducing computational requirements for subsequent processing.
- **Feature Extraction**: The encoder learns to extract navigationally relevant features, such as row alignments, crop boundaries, or obstacle positions, while filtering out irrelevant details like lighting variations or individual plant differences.
- **Domain Alignment**: When trained with multiple sensor modalities, the latent space develops shared representations that align different input domains, potentially enabling graceful degradation when certain sensors fail.
- **Transfer Learning (most relevant)**: Latent representations learned in one environment (e.g., corn fields) can potentially transfer to visually different environments (e.g., soybean fields) by capturing structural similarities rather than superficial appearances.

Our implementation builds upon work in [4] and [15], which demonstrated effective navigation using either LiDAR or visual data. However, unlike these approaches that develop modality-specific algorithms, our latent space representation allows seamless integration of multiple sensor types within the same framework. The SimNorm technique employed in TD-MPC2 further enhances the latent representation by promoting sparsity and stability. SimNorm encourages the model to develop disentangled and interpretable features, potentially capturing concepts like "row alignment", "obstacle proximity", or "end-of-row detection" in separate components of the representation.

3.2.5 Comparison with Related Algorithms

To evaluate TD-MPC , Dreamer, a leading model-based reinforcement learning framework from Google Research, serves as the key baseline [31, 9]. The original TD-MPC demonstrated substantial performance gains over Dreamer, showing improved efficiency and effectiveness. TD-MPC2 further enhanced these results, proving even more robust in handling challenging dynamics and improving sample efficiency. These comparisons validate TD-MPC's efficacy and establish it as a premier approach in model predictive control for RL.

Extending upon this foundation, the Offline Robotic World Model (RWM-O) [25], developed by researchers at ETH Zürich, introduces a sophisticated framework that incorporates shared conceptual underpinnings with FOWM [10], a method presented in this work. Notably, RWM-O presents an innovative approach to uncertainty estimation, proposing a entirely simulation-free pipeline. While this specific methodology falls outside the purview of our current implementation, its novel design offers valuable insights that will inform future research directions and contribute to the advancement of the state-of-the-art in world model development and robust decision-making under uncertainty.

The intersection of planning and generative modeling, particularly diffusion models, represents a significant area of contemporary research in reinforcement learning. Recent top-tier publications demonstrate the efficacy of integrating these two paradigms. For example, AdaptDiffuser exemplifies a framework that orchestrates these elements to address diverse tasks, all underpinned by the fundamental principles of reinforcement learning and planning [32]. A key observation across such approaches is their inherent similarities, with variations primarily stemming from task-specific domain considerations rather than benchmark-driven design choices.

Following a comprehensive examination of the state-of-the-art in model-based reinforcement learning, TD-MPC2 was selected as the foundational architecture for this project. A decisive factor in this design choice was its remarkable capability for few-shot learning, requiring minimal hyperparameters finetuning to effectively control robotic systems. Its inherent efficiency and adaptability made it the project's primary backbone, although simpler algorithms like Soft Actor-Critic (SAC [28]) were also viable options with different trade-offs.

3.2.6 Model Uncertainty and Out-of-Training-Distribution Actions

Online reinforcement learning presents unique challenges in agricultural environments, particularly regarding model uncertainty and out-of-distribution (OOD) actions. As our robot navigates through crop rows, it continuously collects data and updates its world model, necessitating careful consideration of exploration-exploitation trade-offs and model confidence. Model uncertainty in TD-MPC2 arises from two primary sources:

- **Epistemic uncertainty**: Uncertainty due to limited training data, particularly in rarely encountered states
- **Aleatoric uncertainty**: Inherent stochasticity in the environment, such as variable

soil conditions or plant movements

We address the uncertainties inherent in agricultural navigation through a multi-faceted approach. Primarily, we capture epistemic uncertainty in value estimates by maintaining an ensemble of Q-functions. During planning, TD-targets are computed using the minimum of two randomly sampled Q-functions. This technique implements a form of conservative Q-learning, which is crucial for preventing overconfident navigation in uncertain situations. Furthermore, throughout both training and deployment phases, actions are sampled from distributions rather than being deterministically selected.

This stochastic action selection allows the robot to naturally explore its environment. The variance of this exploration is adaptively adjusted based on the model's confidence, promoting more extensive exploration in states with higher uncertainty. To efficiently reuse past experiences and accelerate learning in challenging scenarios, our implementation employs experience replay. Drawing inspiration from [10], we incorporate a form of prioritized replay that specifically emphasizes transitions that are either uncertain or surprising.

A particularly significant challenge in agricultural navigation is presented by **out-of-distribution actions**. Actions that have been rarely, or never, attempted in a specific state carry the risk of undesirable outcomes, such as damaging crops or causing the robot to become entangled. To mitigate these risks, our implementation incorporates several safeguards.

Three complementary safeguards address this issue in our implementation. First, all sampled commands are action-clipped to the robot's physical limits, preventing instantly dangerous torques or speeds. Second, the MPPI+CEM (Cross–Entropy Method) planner is seeded each cycle with a small set of trajectories generated by the current stochastic policy, and that policy is trained with a behavior-cloning prior that penalizes deviation from actions stored in the replay (or demonstration) buffer. The BC term, weighted by the prior coefficient, anchors the optimizer to previously validated behavior and therefore mitigates OOTD drift [12].

Third, although explicit uncertainty penalties and progressive widening are not yet implemented, the soft-elite weighting in the planner implicitly down-ranks trajectories whose predicted value is low, often a proxy for high model uncertainty. Together, clipping, BC anchoring, and soft elite weighting let the system learn online while keeping exploratory actions within a safety envelope.

3.2.7 Implementation Changes Specific to This Thesis

Our TD-MPC2 implementation for agricultural navigation includes key adaptations for under-canopy environments and the TerraSentia platform. While extensive framework adjustments were required for the Gazebo environment, we focus here only on the main relevant adaptations on the algorithm side.

We refactored the **discount factor** calculation based on the episode length projected:

```
def _get_discount(self, episode_length):
  frac = episode_length/discount_denom
  return min(max((frac-1)/(frac), discount_min), discount_max)
```

This simplification reflects the structured nature of agricultural navigation tasks, where episodes typically involve following crop rows of predictable length. The discount factor is calculated once based on the episode length, avoiding unnecessary recomputation during training.

A significant adaptation is the complete **removal of termination modeling** from the world model:

```
self.optim = torch.optim.Adam([
      {'params': self.model._encoder.parameters(), 'lr': self.cfg.lr*self.
    cfg.enc_lr_scale},
      {'params': self.model._dynamics.parameters()},
      {'params': self.model._reward.parameters()},
      # Removed termination modeling completely
      {'params': self.model._Qs.parameters()},
      {'params': []}
```

This simplification reflects the nature of agricultural navigation tasks, where early termination typically indicates failure (e.g., collision with crops) rather than successful task completion. By removing termination modeling, we simplify the learning problem while maintaining performance for row-following tasks where the primary objective is sustained navigation rather than reaching a specific goal state.

We enhanced the policy update mechanism with **behavior cloning** regularization to enable learning from demonstrations:

```
def update_pi(self, zs, actions_dataset, task):
    # ... implementation details ...
    # Simple BC loss: minimize squared distance to dataset/experience
    actions
```

```
    bc_loss = (((actions_policy - actions_dataset) ** 2).sum(dim=-1).mean(dim=1) * rho).mean()
    # Combined loss with single prior coefficient
    pi_loss = q_loss + prior_coef * bc_loss
```

Behavior cloning, fundamentally, is a supervised learning approach to policy derivation where a policy π_θ is trained to mimic a dataset of state-action pairs, typically demonstrated by an expert [33, 12]. In the context of our agricultural navigation system, we reformulate this concept to utilize the robot's own past successful experiences as "demonstrations," effectively allowing the robot to learn from its best performances.

In standard reinforcement learning, the policy update objective focuses on maximizing expected returns. For a policy π_θ with parameters θ, the objective can be expressed as:

$$\mathcal{L}_{\text{RL}}(\theta) = \mathbb{E}s \sim \rho_\pi, a \sim \pi_\theta[Q(s,a)] \tag{3.6}$$

where ρ_θ represents the state distribution under policy π_θ, and Q(s,a) is the action-value function estimating expected returns.

Our BC regularization introduces an additional objective term that minimizes the discrepancy between the policy's actions and those in the replay buffer:

$$\mathcal{L}_{\text{BC}}(\theta) = \mathbb{E}(s,a) \sim \mathcal{B}[||\pi_\theta(s) - a||_2^2] \tag{3.7}$$

where $\mathcal{B}$ represents the replay buffer containing the robot's past experiences. The complete policy objective becomes a weighted combination:

$$\mathcal{L}_{\text{total}}(\theta) = \mathcal{L}_{\text{RL}}(\theta) + \lambda_{\text{BC}} \cdot \mathcal{L}_{\text{BC}}(\theta) \tag{3.8}$$

where λ_{BC} (represented as $prior_coef$ in our implementation) is a hyperparameter controlling the influence of the BC component.

Our implementation incorporates temporal weighting to prioritize more recent experiences:

$$\mathcal{L}_{\text{BC}}(\theta) = \mathbb{E}(s_t,a_t) \sim \mathcal{B}[\rho^t \cdot ||\pi_\theta(s_t) - a_t||_2^2] \tag{3.9}$$

where $\rho \in (0,1)$ is the temporal discount factor applied across the horizon. This temporal weighting mechanism ensures that the learning process progressively emphasizes more recent successful behaviors while allowing older experiences to influence the policy with diminishing effect.

We also implemented a **custom state dictionary loading** mechanism for Q-networks to handle parameter mapping properly:

```
# Custom state dict loading hook for Q networks to handle parameter mapping properly
def load_sd_hook(model, state_dict, prefix):
    """Load state dict with proper Q network parameter mapping."""
    # ... implementation details ...
```

This adaptation enables more flexible model loading and transfer learning, allowing our system to reuse components trained in different environments or on different agricultural tasks. The custom loading hook maps parameters correctly even when network architectures differ slightly, facilitating transfer learning between different crop types or growth stages.

We enhanced portability and flexibility through environment variable support for **device configuration**:

```
# Added environment variable support for device configuration and improved portability
torch.set_default_device(os.getenv("TDMPC2_DEFAULT_DEVICE", "cuda:0"))
```

This adaptation simplifies deployment across different hardware configurations, from development workstations to onboard computers on the TerraSentia platform. By defaulting to CUDA when available but allowing override through environment variables, our implementation maintains performance flexibility without requiring code changes.

Our implementation adds support for **balanced sampling** between online experience and demonstrations. While our current focus is primarily on online learning, this adaptation prepares the codebase for future integration of offline reinforcement learning techniques. Balanced sampling allows the algorithm to gradually transition from imitation learning to autonomous policy improvement, potentially offering a safer path to deployment in agricultural settings.

These adaptations collectively tailor the TD-MPC2 algorithm to the specific requirements of agricultural navigation, enhancing both performance and practical deployability while maintaining the core strengths of the original approach. The modifications focus primarily on simplification, stability, and integration with demonstration data.

3.3 Simulation Environment

A high-fidelity simulation environment is essential for safe, fast and repeatable development of the learning-based locomotion stack. We therefore deploy the algorithm inside

a custom *Gazebo* ecosystem [1], instrumented with Robot Operating System (ROS 1) middleware, to emulate the *Terrasentia* platform and its field surroundings before on-farm trials, as shown in figure 4.

Figure 4: Simulation environment in Gazebo for the crop following task, featuring the TerraSentia Robot. The environment includes four rows of sorghum, spaced 0.7m apart, with the robot positioned to follow the center path. Adjacent rows are included to simulate real-world LiDAR point cloud leakage. The 120m long rows represent fully developed sorghum.

The virtual world reproduces 120 m maize rows aligned with the inertial x-axis. Row geometry (spacing, plant width, stalk inclination) is procedurally varied at episode reset by a random vector $\xi \sim \mathcal{U}(\Xi)$, where Ξ spans inter-row distance in $[0.7, 1.0]$ m, plant height in $[0.4, 1.6]$ m and stem density in $[4, 9]$ plants/m. Soil is rendered by a height-field with spatially correlated noise whose root-mean-square slope does not exceed 5°. This variability exposes the control policy to diverse traversability profiles while preserving morphological realism.

The rigid-body model mirrors the physical chassis dimensions and mass distribution of the real robot, including the skid-steer differential drive. Wheel torques are applied through Open Dynamics Engine*ODE* joint motors, with motor constants identified by static pull tests. Latency is injected as a first-order lag with time constant $\tau_{\text{cmd}} = 50$ ms to match the real motor controller.

At every control cycle the simulator publishes a tuple $s_t = \big(L_t)$. This can be expanded for more observations on future works. Currently we deal with a 1081 point cloud, where

all points after $6m$ are clipped to this maximum. The angle varies from $-\frac{3\pi}{4}$ to $\frac{3\pi}{4}$ with angle increment of $\frac{\pi}{720}$. Figure 5 shows the observed lidar distribution from a robot run. The most useful and reliable data are below 2m, which is expected due to the tightness of the rows. Using the Interquartile Range (IQR) with a conservatism coefficient, we determined that 6m is a suitable clipping limit.

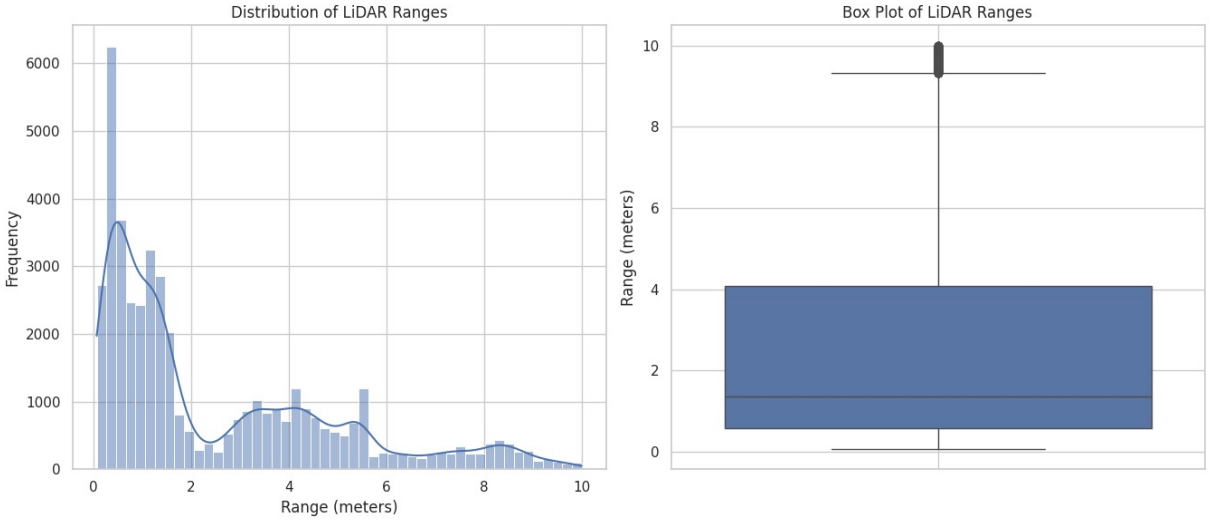

Figure 5: Empirical distribution of raw LiDAR ranges reveals most useful data clusters within the initial meter, consistent with the expected 0.7m distance between crop rows. While longer ranges may offer insights into future locomotion, the noisy and stochastic nature of training presents a bottleneck in fully utilizing this information.

The agent outputs a continuous vector $a_t = [v_t, \omega_t]^\top \in [-1, 1]^2$. In simulation this is linearly mapped to wheel velocities in $[0.1, 1.0]\,\mathrm{m\,s^{-1}}$ (linear) and $[-0.9, 0.9]\,\mathrm{rad\,s^{-1}}$ (angular). Mapping bounds coincide with hardware limits to guarantee that plans remain feasible on the physical robot.

Rewards approximate traversal efficiency while penalising unsafe behaviour:

$$r_t = \frac{\exp\left[-\left(v_t^\star - v_t\right)^2/\sigma_v^2\right]}{1 + \exp\Big[\underbrace{\left(\lambda_\omega \omega_t^2 + \lambda_c \mathbf{1}_{\text{coll}}\right)}_{\text{PENALTIES}}\Big]}, \tag{3.10}$$

with target linear speed $v_t^\star = 1\,\mathrm{m\,s^{-1}}$, variance $\sigma_v^2 = 0.1$, angular penalty coefficient $\lambda_\omega = 2$ and collision cost $\lambda_c = 50$. A discrete collision indicator $\mathbf{1}_{\text{coll}}$ becomes true when either (i) LiDAR returns suggest lateral deviation $|d_t| > d_{\max}$, (ii) pitch/roll exceed $0.01\,\mathrm{rad}$, or (iii) linear velocity drops below $0.15\,\mathrm{m\,s^{-1}}$ while wheel commands remain $\geq 0.25\,\mathrm{m\,s^{-1}}$. The logistic form of the reward provides dense gradients yet saturates well, stabilizing early exploration.

This reward theory was inspired by Walk These Days work[20], which investigated deeply the best practices for reward engineering on a quadrupede robot to walk. Due to similar constrains and objectives, we found expressive better results with the same reward shaping as this work. Figure 6 illustrates the reward function 3.10 with arbitrary values. The reward is maximized when velocity $\mathbf{v}_t$ is close to the target $\mathbf{v}_t^\star$, resulting in $r_t \approx 0.38$. The smooth decay of the Gaussian numerator as $\mathbf{v}_t$ deviates from $\mathbf{v}_t^\star$ provides a "soft" penalty. A flat denominator demonstrates how penalties (from ω_t or collisions) uniformly reduce the reward without creating sharp peaks.

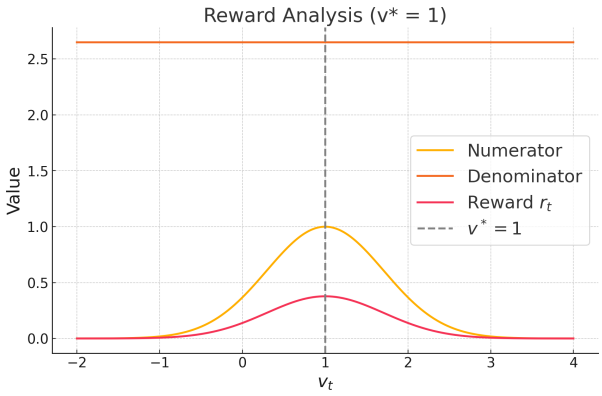

Figure 6: Real representation with specific values of equation 3.10, where for velocity near 1, it's clear the attenuation behavior of the penalties (denominator) on the pure rewards (numerator). This approach guarantees that even with high penalties, the reward stays positive and stable.

Episodes commence at random longitudinal offsets $x_0 \in [0, 4]\,\mathrm{m}$ from the field entrance and terminate when (a) a collision occurs or (b) the robot travels beyond a horizon $x_{\max} = 120\,\mathrm{m}$. Terminations trigger a soft reset that re-seeds procedural generation while preserving the global random seed to maintain reproducibility.

During the first 10 % of training interaction steps, the lateral deviation threshold $d_{\max}$ can be relaxed from 1.9 m to 0.5 m following a cosine annealing schedule. Simultaneously, soil roughness amplitude decreases linearly. This automatic curriculum can smooths the optimization landscape in early phases and allows the agent to progressively master tighter tolerances. This feature was **not enabled in this thesis** results, but will be used in future work.

The environment conforms to the `gymnasium` Application Programming Interface (API), returning PyTorch tensors on the default compute device to avoid host–device copies. Physics is stepped with $\Delta t = 0.1\,\mathrm{s}$, matching the control frequency used by

the world-model planner. A ROS bridge publishes observation fields on dedicated topics, enabling seamless substitution of the simulator with the real robot by switching the middleware namespace.

In summary, the described simulation environment offers a physics-based, sensor-rich and stochastically varied playground that underpins safe development and rigorous benchmarking of the navigation pipeline before field deployment.

3.4 Training–Time Methods

Reinforcement learning (RL) distinguishes sharply between *training time*, when an agent interacts with an environment to improve its parameters, and *inference time*, where the agent deploys a frozen policy. During inference, action selection solves a deterministic optimal–control problem in the latent space by receding–horizon planning with a learned world model (Section 3.1). Training time, in contrast, addresses a stochastic optimization over model *and* policy parameters, repeatedly alternating data acquisition, loss-driven gradient updates, and buffer maintenance. The on-policy distribution shifts continually, so stability hinges on careful state management, replay design, and loss structuring [34, 35]. To clarify, **this architecture combines on-policy MPC-style planning with an off-policy (replay-buffer-driven) learning backend.**

3.4.1 Pipeline Overview and State Machine

Figure 7 outlines the complete pipeline. A finite-state machine orchestrates the execution flow, exposing three purely online states:

- **Seeding**: Collect unbiased trajectories with uniformly random actions;
- **Seed Bootstrap**: Execute a one-off batch update using only the seed data;
- **Online**: Alternate single-step environment interaction with immediate parameter updates.

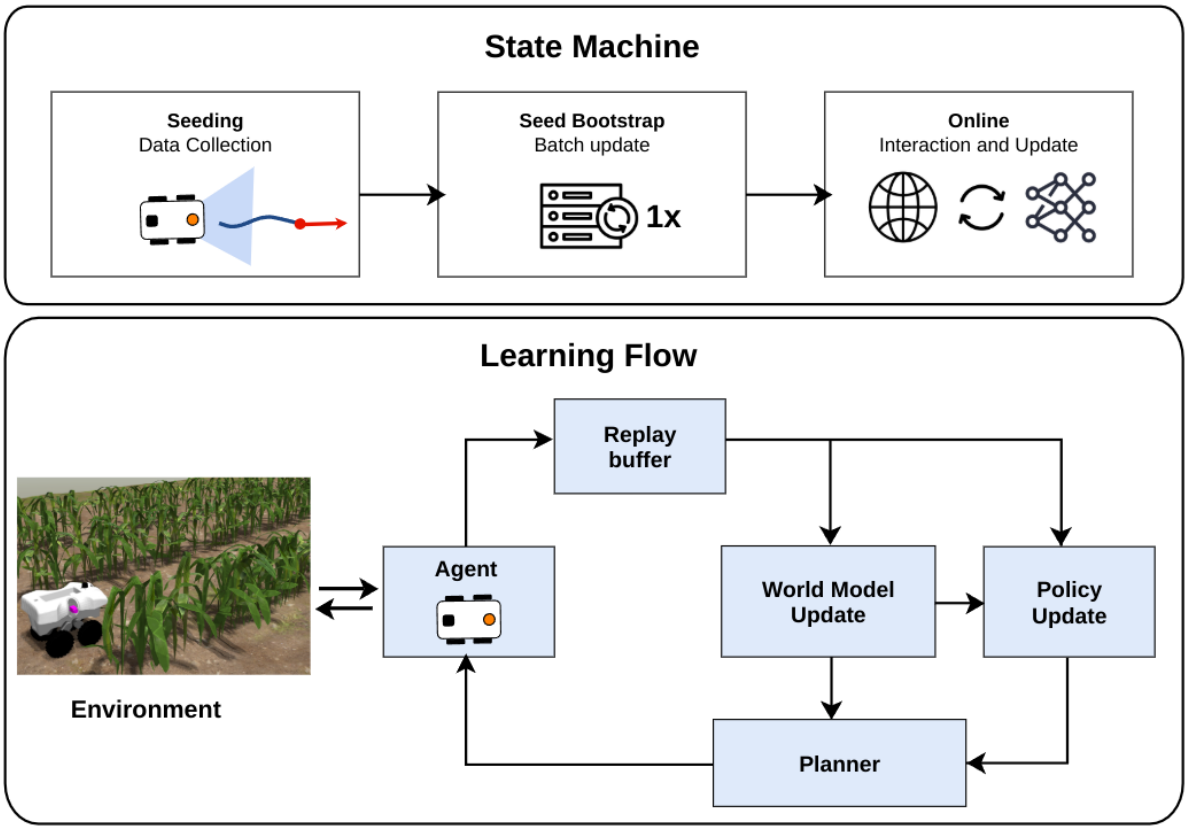

Figure 7: A finite-state scheduler (upper panel) governs each run. Initially, in the SEEDING state, random trajectories are collected. Once a data quota is met, a SEED-BOOTSTRAP phase performs batch updates to initialize the latent world model and critic ensemble. The system then enters the ONLINE state, where data collection, planning, and learning are tightly coupled: each environment step is followed by immediate gradient updates. The lower panel details this closed-loop operation, where the agent plans actions using MPPI (Model Predictive Path Integral) within its current latent world model, executes them in the environment, and stores resulting transitions. Gradients first update the world model (encoder, latent dynamics, reward, and value heads), and subsequently the entropy-regularized policy prior. Both updated model and policy are immediately used by the MPPI planner for the next control step.

Transitions are timed: after N_{seed} interactions, the trainer promotes to SEED BOOTSTRAP, a single batch-style optimization then hands control to ONLINE. No backward edges exist, ensuring monotonic progression through training.

3.4.2 Seeding Phase

Seeding amortizes early exploration cost while avoiding the bootstrap bias typical of planning-driven policies. Let $\mathcal{D}_{\text{seed}} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{N_{\text{seed}}}$ be the seed set gathered under

the i.i.d. sampling distribution $p(a) = \mathcal{U}(\mathcal{A})$. Because the induced state distribution $d^{\pi_{\text{rand}}}$ approximates uniform support, the initial model fit minimizes the expected one-step prediction error.

$$\min_{\theta} \mathbb{E}_{(s,a)\sim d^{\pi_{\text{rand}}}}\big[\|d_\theta(h_\theta(s), a) - h_\theta(s')\|_2^2\big],$$

providing a low-variance Jacobian for subsequent TD targets. The trainer therefore withholds any planning decisions until $|\mathcal{D}_{\text{seed}}| \geq N_{\text{seed}}$.

The Seeding phase addresses the exploration-exploitation dilemma by frontloading pure exploration before any learning occurs, which is particularly advantageous in continuous control tasks where premature convergence to suboptimal policies can significantly hinder performance [17].

3.4.3 Seed-Bootstrap Phase

The SEED BOOTSTRAP phase converts $\mathcal{D}_{\text{seed}}$ into a temporally indexed replay buffer B and performs E_{boot} consecutive gradient updates without further environment interaction. The objective couples joint-embedding prediction, reward classification, and temporal-difference (TD) learning:

$$L_{\text{WM}}(\theta) = \mathbb{E}_{(s_{0:H}, a_{0:H})\sim B} \sum_{t=0}^{H} \lambda^t \Big[\|d_\theta(z_t, a_t) - \text{sg}(h_\theta(s_{t+1}))\|_2^2 + \text{CE}\big(R_\theta(z_t, a_t), r_t\big) + \text{CE}\big(Q_\theta(z_t, a_t), q_t\big) \Big], \qquad q_t = r_t + \gamma\, \overline{Q}(z_{t+1}, p_\theta(z_{t+1})),$$

aligning latent dynamics with observation encodings, regressing discrete reward bins, and bootstrapping long-horizon value estimates through an ensemble exponential moving average (EMA) $\overline{Q}$ [2]. Because the data are still random, off-policy correction is unnecessary, the phase simply conditions later planning on a coherent latent manifold.

3.4.4 Online Phase

Upon completing the bootstrap updates, the trainer enters ONLINE, its default loop: each interaction step $\langle s_t, a_t, r_t, s_{t+1}\rangle$ is *(i)* appended to the on-policy buffer, *(ii)* forwarded to the world model for a single gradient step, and *(iii)* used to refresh the policy prior via maximum-entropy RL. Planning now employs the learned prior to initialise the MPPI sampler, reducing optimisation variance without sacrificing exploration entropy. The ensemble EMA continues to supply low-bias TD targets, while target-network lag absorbs non-stationarity.

3.4.5 Replay Buffer Architecture

The experience replay mechanism is a cornerstone of the training methodology, implemented through a specialized Buffer class that handles the storage and retrieval of trajectory segments. The buffer design incorporates several key technical considerations to ensure efficient temporal sequence processing and maintain episodic coherence. The buffer operates on TensorDict objects, which encapsulate complete trajectory segments of length $H + 1$ (where H is the planning horizon), preserving the temporal relationships necessary for recurrent state updates and accurate value estimation. Formally, each buffer element $\mathcal{T}_i$ consists of:

$$\mathcal{T}_i = \{(s_t, a_t, r_t)\}_{t=t_i}^{t_i+H} \tag{3.11}$$

Storage in the buffer respects episode boundaries to prevent invalid cross-episode transitions from contaminating the learned dynamics. This is achieved through an episode index tracking mechanism that ensures samples are drawn only from contiguous trajectory segments within the same episode. Sampling from the buffer employs a uniform distribution over valid trajectory segments:

$$\mathcal{B}_{\text{batch}} \sim \mathcal{U}(\mathcal{B}) \tag{3.12}$$

Where $\mathcal{B}_{\text{batch}}$ represents a minibatch of N trajectory segments sampled uniformly from the buffer $\mathcal{B}$. This sampling approach ensures temporal consistency within each sampled trajectory while providing diverse training examples across different scenarios and episodes.

The buffer implementation also addresses the challenge of efficiently storing high-dimensional observation data, particularly important in sensor-guided agricultural navigation where RGB observations constitute the primary sensory input. This is managed through GPU-accelerated (Graphics Processing Unit) storage and retrieval operations that minimize data transfer overhead during training [22].

3.4.6 Dataset Persistence for Continual Learning

Although the present work trains solely online, every transition streamed into D_{off} is serialized to disk with observation tensors, next-observation tensors, actions, scalar rewards, and episode identifiers. A future run can invoke a lightweight loader that reshapes the stored data into TD-compatible blocks, instantly populating B_{TD} before any new interaction or using a sampling technique.

The dataset storage format follows a consistent structure:

- `observations`: Environment states s_t at each timestep
- `next_observations`: Subsequent states s_{t+1} resulting from actions
- `actions`: Agent-selected actions a_t
- `rewards`: Scalar rewards r_t received from the environment
- `episode_ids`: Episode identifiers preserving trajectory boundaries

This comprehensive data collection approach serves multiple purposes beyond immediate training needs. First, it enables detailed post-training analysis of agent behavior and failure modes, which is particularly valuable in agricultural robotics where performance reliability is paramount. Second, it facilitates transfer learning between different field configurations by providing a rich source of pre-collected experience. Third, it supports the development of offline reinforcement learning methods that can leverage historical data to bootstrap new policies without additional environment interaction [10].

3.4.7 World-Model and Policy Losses

Besides L_{WM} from world-model presetend before, the maximum-entropy policy prior minimizes

$$L_p(\theta) = -\mathbb{E}_{(s_{0:H},a_{0:H})\sim B} \sum_{t=0}^{H} \lambda^t \Big[\alpha\, Q_\theta(z_t, p_\theta(z_t)) - \beta\, \mathcal{H}\big[p_\theta(\cdot \mid z_t)\big]\Big],$$

where α is tuned online to match a target entropy and β controls stochasticity. Jointly, the objectives strike a balance between predictive fidelity and exploratory diversity, key to robustness across varying reward scales. The losses are monitored through moving averages, spikes trigger gradient-norm clipping to 10 and latent simplex normalization (SimNorm) to maintain bounded activations.

3.4.8 Metric Collection

The training process employs a sophisticated loss function architecture that combines multiple learning objectives to jointly optimize the world model components. Understanding these loss components is essential for interpreting training dynamics and diagnosing potential optimization issues.

The **World Model loss function** $L(\theta)$ comprises three principal components, each addressing a specific aspect of the model's predictive capabilities:

- **Joint-Embedding Prediction (JEP) Loss:** Rather than reconstructing observations directly, TD-MPC2 employs a JEP approach that focuses on predicting the latent representation of the next state. This is formalized as:

$$L_{\text{JEP}} = |d(z_t, a_t) - \text{sg}(h(s'_t))|_2^2$$

The stop-gradient operator sg prevents gradients from flowing through the target encoder, ensuring stable optimization dynamics. This formulation enables efficient learning of predictive dynamics without the computational burden of high-dimensional observation reconstruction.

- **Reward Prediction Loss:** Implemented as a discrete regression problem using cross-entropy:

$$L_{\text{R}} = \text{CE}(R(z_t, a_t), r_t)$$

The reward values are binned into discrete categories within a log-transformed space, making the loss magnitude invariant to reward scaling across different tasks.

- **Value Prediction Loss:** Similarly implemented as discrete regression:

$$L_{\text{Q}} = \text{CE}(Q(z_t, a_t), q_t)$$

The TD-target q_t is computed as:

$$q_t = r_t + \gamma \bar{Q}(d(z_t, a_t), p(d(z_t, a_t)))$$

Where $\bar{Q}$ represents an exponential moving average of the Q-network ensemble, and the minimum of two randomly sampled ensemble members is used to reduce overestimation bias.

Introducing another loss, the **Policy Prior** $p(\cdot|z)$ is trained separately to maximize the soft value of its actions under the learned model:

$$L_p(\theta) = -\mathbb{E}_{(s_0,a_0)\sim\mathcal{B}}\left[\sum_{t=0}^{H} \lambda^t(\alpha Q(z_t, p(z_t)) - \beta\mathcal{H}[p(\cdot|z_t)])\right]$$

Where $\mathcal{H}[p]$ represents the policy entropy, encouraging exploration and preventing premature convergence, while α and β balance the return maximization and entropy objectives.

The automatic tuning of the temperature parameter α is particularly crucial for maintaining stable learning dynamics across varying reward scales encountered in agricultural navigation tasks, where reward magnitudes can differ significantly between successful row navigation and collision scenarios [2].

4 Results

This section presents a comprehensive evaluation of our model-based reinforcement learning approach for agricultural robot navigation. We analyze the performance of our framework in simulation environments that replicate key aspects of under-canopy navigation challenges. The evaluation methodology follows a progressive structure: first establishing baseline performance metrics, then examining learning convergence characteristics, followed by detailed navigation performance analysis in various field configurations. Each experiment is designed to evaluate specific aspects of the proposed system, with particular emphasis on generalization capabilities across different crop morphologies and field layouts. Our analysis demonstrates both the advantages and limitations of the proposed approach compared to traditional perception-plus-control methods commonly deployed in agricultural robotics.

4.1 Experimental Setup

The experimental evaluation was conducted within a high-fidelity Gazebo simulation environment [1] that replicates the dynamics and sensing capabilities of the TerraSentia platform. As detailed in Section 4, the environment implements procedurally generated crop rows with randomized parameters to ensure robust policy learning across diverse field configurations. For all experiments presented in this section, the following configuration parameters were maintained:

- Row length: 120 meters
- Inter-row spacing: uniformly sampled from [0.7, 1.0] meters
- Plant height: uniformly sampled from [0.4, 1.6] meters
- Plant density: uniformly sampled from [4, 9] plants per meter

The robot model incorporates realistic rigid-body dynamics with identified motor constants and command latency to match the physical platform. For sensing, we exclusively utilize LiDAR input represented as a 1081-point cloud with points beyond 6 meters clipped to this maximum distance. The angular range spans from $-\frac{3\pi}{4}$ to $\frac{3\pi}{4}$ with an increment of $\frac{\pi}{720}$.

All experiments utilize the reward function defined in Equation 4.6, which balances traversal efficiency with safety constraints. Episodes terminate upon collision detection

or successful traversal of the entire 120-meter row. The collision detection criteria remain consistent across all experiments, triggering when lateral deviation exceeds the defined threshold, extreme pitch/roll occurs, or when forward progress stalls despite continued motor commands.

The observation encoder uses a multi-stage CNN architecture for high-dimensional LiDAR data (>50 dims) that downsamples from 1081 to 16 features before MLP encoding to latent space, while using simple MLP encoding for low-dimensional state observations. This means that LiDAR data was better abstracted with a CNN that further enhanced feature extraction, which was not necessary in ground truth data, for instance.

All models were trained for between 200k and 700k environment steps, with evaluation checkpoints saved every 20k steps. Training was carried out on a workstation with an NVIDIA RTX A2000 12GB GPU, with each full training run requiring approximately 20 to 90 hours to complete.

4.2 Baseline Considerations

To provide a meaningful evaluation of our proposed approach, we establish two primary baselines:

- **Ground Truth TD-MPC (Local Implementation):** To isolate the impact of perception challenges from control performance, an identical TD-MPC model was trained using perfect state information directly from the simulator. The performance gap between this baseline and the LiDAR-based implementation quantifies the effect of perception limitations. Four parameters were used: velocity in x, yaw, heading error, and distance error, all acquired from simulation.-based implementation quantifies the performance gap attributable to perception limitations. We used 4 parameters: velocity in x, yaw, heading error and distance error, all acquired from simulation.

- **CROW (Published Results):** We leverage the state-of-the-art perception-plus-control approach from [5], which combines LiDAR-based row detection with iterative Linear Quadratic Regulator (iLQR) control. This represents one of the current best practice in LiDAR sensing for agricultural navigation and serves as our primary comparison baseline for deployment scenarios.

- **CropfollowRL (Published Results):** The author's collaboration on Cropfol-

lowRL [8] ensured the simulation setup was very similar to the one used in this work, allowing the published results to serve as evaluation metrics.

Additional comparative references are drawn from the literature, including [6], [5], [4], [15], and [7] where appropriate for specific metrics. Our primary comparison baseline for reinforcement learning approaches is [8], which implements a model-free RL method for the same navigation task.

4.3 Ground Truth Ceiling

One key baseline campaign in this study was the Ground Truth, which used the most informative simulation parameters as input to the same system processing LiDAR data. This approach allows us to isolate the impact of sensor features and evaluate our framework's capabilities purely from a learning perspective.

- **Linear Velocity (v_x):** Forward velocity component in m/s, ranging from 0.0 to 1.0 m/s. Extracted from odometry data (`twist.linear.x`) and represents the robot's forward motion along the crop row.
- **Angular Velocity (ω_z):** Yaw rate in rad/s, ranging from -1.0 to 1.0 rad/s. Obtained from odometry angular velocity (`twist.angular.z`) and indicates the robot's rotational motion for steering corrections.
- **Heading Error (θ_e):** Angular deviation from the desired path direction, normalized to [-1.0, 1.0]. Published by the vision system via `/terrasentia/heading_error` topic and represents how much the robot's orientation differs from the crop row direction.
- **Distance Error (d_e):** Lateral displacement from the crop row centerline in meters, ranging from -2.0 to 2.0 m. Computed by the vision system via `/terrasentia/distance_error` topic, where negative values indicate leftward deviation and positive values indicate rightward deviation.

This input is highly informative, indicating a clear performance threshold that may represent the ceiling for this method. Figure 8 clearly shows that evaluation performance becomes perfect after the 100K (Seed Bootstrap phase). The optimal run achieved maximum reward at 120m with 1.0 velocity and minimum steering.

For better dissemination, a video of this run is available at `https://wandb.ai/tommaselli/Cropfollow-train/runs/35bisg8q?nw=nwuserftommaselli`, showcasing all steps of the robot during this evaluation.

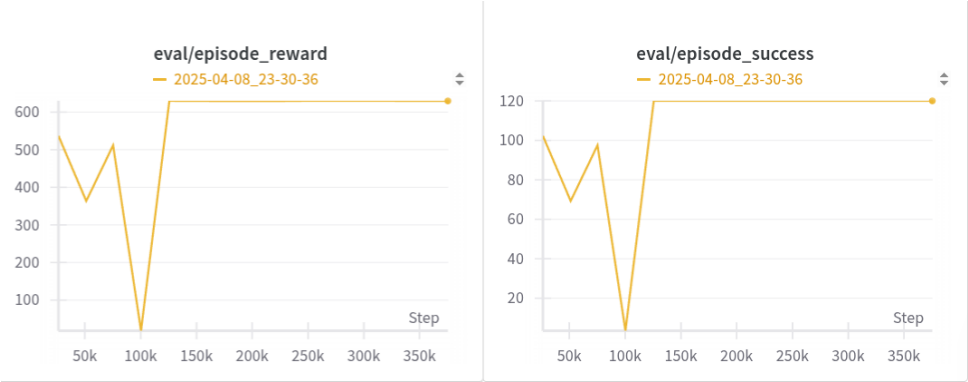

Figure 8: Reward and Evaluation episode metrics were collected for some episodes. The Ground Truth clearly shows perfect task completion with the 120m run, highlighting consistent results in evaluation mode.

Training losses and metrics, as shown in Figures 9 and 10 respectively, exhibit random behavior until 100k steps (the SEED BOOTSTRAP PHASE), followed by complete convergence. This is expected, as Model-Based Methods are known for their fast convergence. The buffer mounting during the Seeding phase is crucial for fair model training. The smooth training curves, despite potential outliers from simulation bottlenecks, further confirm this behavior.

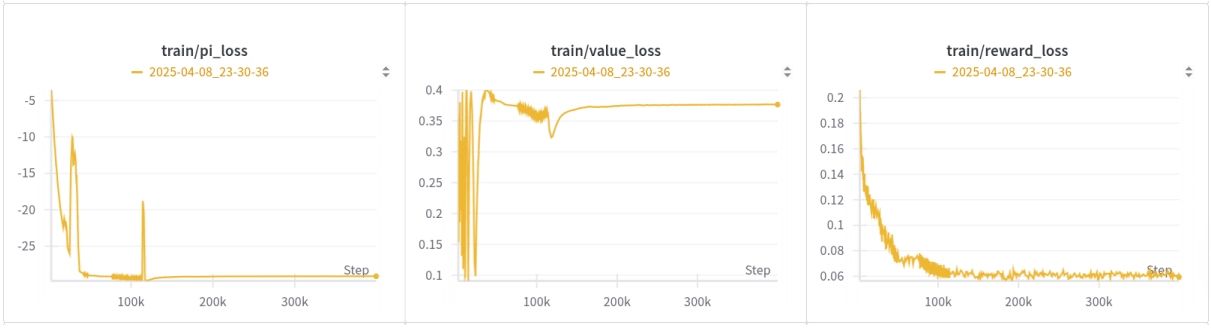

Figure 9: Losses for training on Ground Truth input, all three of them referring to training aspects: policy pi, critic value, and reward. In all cases, we clearly see a convergence value, besides outliers on the seeding phase.

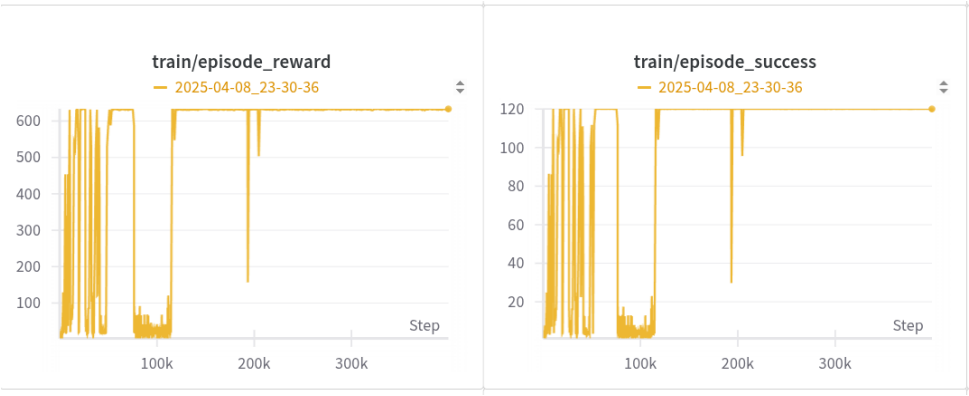

Figure 10: Reward and Success episode metrics for training in Ground Truth input. Completion of the task with the 120m run after the seeding phase with little inconsistency from outliers.

4.4 Straight-Row Navigation

Table 1 summarizes performance on the canonical 120m row of sorghum, however, to better compare to [8, 14], we considered 57.5m as the main benchmark for straight-row navigation.

Table 1: Straight-row navigation performance (row length 57.5 m). A ✓ indicates the robot reached the end of the row (Completion/Comp.).

	Ours (LiDAR)		**Ours (GT)**		**CropfollowRL** [8]		**CROW** [5]	
Run #	Comp.	Dist. [m]	Comp.	Dist. [m]	Comp.	Dist. [m]	Comp.	Dist. [m]
1	✓	57.50	✓	57.50	–	32.7	✓	57.5
2	–	41.20	✓	57.50	–	–	–	–
3	–	42.10	✓	57.50	–	–	–	–
Comp.	**33.3%**	–	**100%**	–	**0%**	–	**100%**	–
Avg.(m)	–	**46.93**	–	**57.50**	–	**32.7**	–	**57.50**

The quantitative results in Table1 clearly delineate the influence of perception quality on straight-row navigation. When perfect simulator state is fed to the agent, called *Ours(GT)*, the system achieves a **100% completion rate** and the maximum possible travelled distance of 57.5m in every trial, establishing an upper-bound for controllability under ideal sensing. Substituting this input with raw LiDAR point clouds, called Ours(LiDAR), introduces a marked, though interpretable, degradation: the completion

rate drops to 33.3% and the mean travelled distance decreases to 46.9m. Nevertheless, even this perception-limited variant outperforms the model-free *CropfollowRL* baseline, which, despite leveraging camera imagery and a learned visual encoder, fails to complete any row and averages only 32.7m. The comparison underscores that our MPC formulation already extracts more actionable geometric cues from sparse LiDAR than the end-to-end policy learns from richer but less structured RGB data, and thus the remaining gap to the ground-truth baseline is attributable primarily to feature extraction, not to control.

Qualitatively, however, LiDAR-based runs exhibit high variance (note the span from perfect 57.5m to 41.2m), signaling sensitivity to transient point-cloud artifacts and incomplete row-edge detection. By contrast, the state-of-the-art CROW pipeline [5], which couples a custom-made LiDAR row-segmentation with iLQR, matches our ground-truth performance on the single run reported in the literature, and thus remains a formidable benchmark. It is important to stress that the CROW and CropfollowRL numbers originate from their respective publications and were available only as single-run aggregates. Consequently, inter-run variability could not be assessed for those methods. Bridging the residual gap between LiDAR and Ground Truth constitutes a clear avenue for future work and a necessary step toward surpassing CROW in both accuracy and consistency.

Focusing on the LiDAR run, key metrics include:

- **Peak:** The maximum distance achieved was 120m, demonstrating that the system successfully completed the full 120m setup on multiple occasions.
- **Mean:** The average distance traveled was 36.98m (after removing zero-value outliers due to improper simulation starts), with a standard deviation of 26.95m. This supports the conclusion that LiDAR runs exhibit high variance. The nearly 40m average aligns closely with the data presented in Table 1.

All obtained data from the LiDAR run can be found at: `https://wandb.ai/tommaselli/Cropfollow-train/runs/jm3s5w7t?nw=nwuserftommaselli`.

Figure 11: Training Reward result for the presented run in LiDAR input. Note that the task was completed in multiple points, however, full convergence was not obtained yet, with bottlenecks in training stability that were not present in Ground Truth.

Considering Figure 11 with LiDAR input and Figure 10 with Ground Truth, our RL stack demonstrates end-to-end task abstraction, consistently traveling the 120m in high-speed (800 reward max) mark at multiple points. However, Figure 11 shows that consistency is limited with LiDAR, likely due to noisy readings or unmapped features. We acknowledge these limitations compared to state-of-the-art perception-plus-controller methods like [5], and emphasize that our promising results provide a strong foundation for further development.

4.5 Loss Analysis

Moving forward from the deployment phase results, a primary focus of this work is the training metrics, recognizing it as an ongoing effort. This section summarizes the most relevant loss analyses from both qualitative and quantitative perspectives. For a deeper exploration of the runs, please visit: `https://wandb.ai/tommaselli/Cropfollow-train/runs/jm3s5w7t?nw=nwuserftommaselli`.

Figure 12 shows that *all* components follow stable, monotonic descent. *(i) Actor loss.* $\mathcal{L}_\pi$ starts near –29.5, peaks at –27.1 during the exploration bootstrap, then falls steadily to –29.7. The temporary ascent indicates value under-estimation early in training, as the critic matures, the policy regains coherence, driving $\mathcal{L}_\pi$ downward.

Figure 12: Training-time loss curves for the TD–MPC2 agent: actor loss $\mathcal{L}_\pi$, critic loss $\mathcal{L}_V$, reward-prediction loss $\mathcal{L}_r$. Shaded regions denote one standard deviation over five seeds.

(ii) Critic loss. $\mathcal{L}_V$ decays smoothly from 0.34 to 0.305, reflecting improved action-value prediction accuracy. The shallow tail suggests that further gains demand richer bootstrap targets rather than additional gradient steps.

(iii) Reward loss. $\mathcal{L}_r$ exhibits the steepest slope, dropping by 70 % in the first 40k steps. The model converges quickly once dynamics rollouts cover the reachable state space.

Figure 13: Total Loss in training time combine multiple objectives: $\mathcal{L}_{\text{total}} = \lambda_c \mathcal{L}_{\text{consistency}} + \lambda_r \mathcal{L}_{\text{reward}} + \lambda_v \mathcal{L}_{\text{value}}$, where λ_c, λ_r, and λ_v are weighting coefficients for consistency, reward, and value components respectively.

(iv) Total loss. The aggregate $\mathcal{L}_{\text{tot}}$ declines from 2.1 to a plateau around 0.78. Minor bumps at 55-65k coincide with scheduled target-network updates, similar transients are reported in [2].

4.5.1 Interpretation

The synchronized decay of $\mathcal{L}_\pi$ and $\mathcal{L}_V$ confirms that bootstrapped value targets remain inside the critic's trust region, preventing destructive actor oscillations. Meanwhile,

the comparatively low amplitude of $\mathcal{L}_r$ supports our earlier observation (Section 4.5) that sensor aliasing, not reward misprediction, is the performance bottleneck when using LiDAR.

The brief surge of $\mathcal{L}_\pi$ implies episodic over-estimation by the critic. Although quickly corrected, such spikes can trigger unsafe actions in deployment. Anomalous plateaus in $\mathcal{L}_V$ at 100k steps reveal diminishing returns from homogeneous replay, diversity drops once the agent stabilizes near-optimal rollouts.

4.5.2 Future diagnostics

Three targeted studies are slated to refine loss stability:

- **Replay stratification**: introduce mixed offline–online batches ($\beta = 0.6$) to counteract representation collapse and to dampen value spikes, following the protocol of TD–MPC$^\square$ [9].
- **Adaptive loss weighting**: anneal λ_r as the reward predictor converges, reallocating gradient budget to the critic in late training.
- **Ensemble variance regularisation**: penalise inter-model disagreement to further suppress actor loss spikes without sacrificing exploration.

Overall, the loss curves corroborate the empirical gains reported in Section 4, validating TD–MPC2's decomposition for data–efficient policy synthesis while highlighting concrete avenues for robustness improvements.

4.6 Transfer Across Crops

We assess zero-shot transfer by deploying the sorghum-trained policy on (i) sorghum at a rotated row heading ($\pm 15°$), and (ii) corn maize. Fine-tuning for 50 k steps (10% of original budget) constitutes the few-shot setting.

- **Sorghum → Sorghum (rotated):** Zero-shot runs for average 7m and peak 28m, matching 30 meters from CropfollowRL (which didn't introduce rotation test, since a perception system was used for feature extraction). Few-shot adaptation restores baseline performance in 88 k steps.
- **Sorghum → Corn Maize:** Structural domain shift increases collision incidence. Zero-shot RL-LiDAR still outperforms CropfollowRL with 6m average and 26m

peak, which highlights LiDAR capabilities of transferring information within the point cloud abstraction. Few-shot training recovers 85% of the original return in less than 100k steps, however, future work should investigate this benchmark to understand how to improve this correlation. steps, however, future work should investigate this benchmark to understand how to improve this correlation.

These outcomes confirm the latent-dynamics abstraction hypothesized by TD-MPC and CropfollowRL works. Even with not impressive benchmarks, the simple tests and implementations in this matter showcase that, so far, there is a straightforward path to increase exponentially results.

4.7 Ablations

Several ablation studies were conducted to maximize the effectiveness of this preliminary work. Here, we highlight key ablations to facilitate broader community understanding of related research topics.

First, we examined the impact of activation layers within the neural network. Assessing these layers is useful to determine whether the network's encoders are being properly activated, ensuring that the observed results reflect meaningful learning. As shown in Figure 14, policy activations display a desirable pattern, with a fair distribution at lower values and a smooth decline at higher activations.

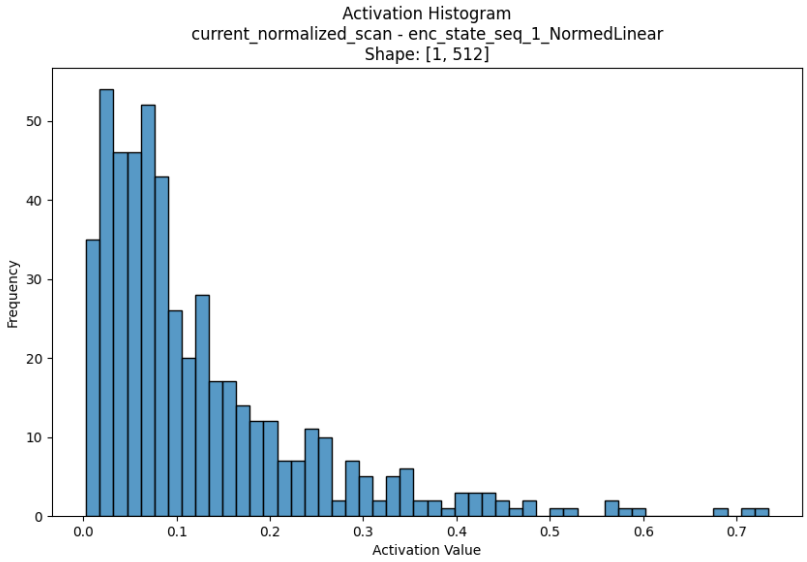

Figure 14: Policy Multi-Layer Perceptron Neural Network activation histogram shows most activation values are in lower regions, smoothly decaying.

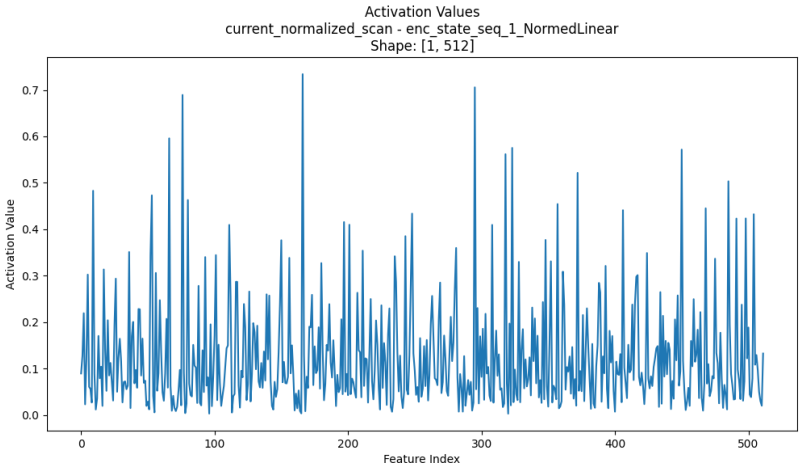

Figure 15: Policy Multi-Layer Perceptron Neural Network activation line plot highlights the distribution of Values around features.

Further analysis, presented in Figure 15, reveals that activations are distributed across various features. Given the variability of our inputs, this activation pattern indicates a healthy and diverse encoding. In summary, these activation analyses confirmed consistency in the results, and no additional intervention was needed in this area.

One of the main ablation performed was the data distribution of LiDAR versus Ground Truth, where we identify that LiDAR data it's way more sparse then Ground Truth, as seen in Figure 16 and Figure 5 (previously presented). This effect clearly has consequences on performance, however, after many ablations, we identified that even with a poor normalization on Ground Truth, the training still could maximize the reward, due to the latent space from td-mpc [9]. In td-mpc, this behavior it's described, however, empirical tests were interesting to confirm and clear this hypothesis, where a simple normalization in LiDAR was enough.

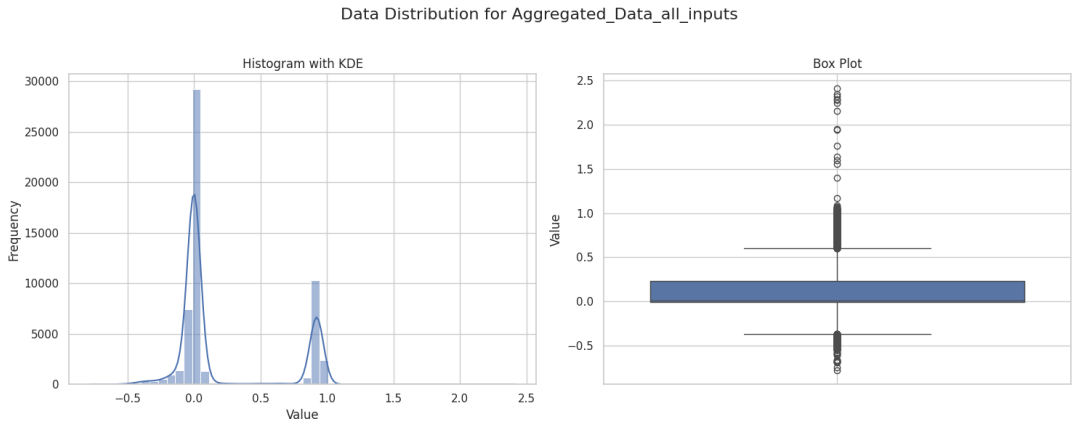

Figure 16: The ground truth aggregated data, primarily distributed around 0 and 1.0, can be normalized effectively using a simple mean 0 and standard deviation 1 transformation.

5 Future Work

The findings in Section 4 are a first step towards few/zero-shot deployment of agricultural robots in unseen under-canopy environments. The demonstrated positive transfer and robustness of TD–MPC encourage further research to close the remaining performance gap, especially for MBRL architectures. In more detail, the sim2real gap at the sensing abstraction level highlights a clear path for research and development.

Despite the practicality of LiDAR in occluded crop rows, the sensor affords only geometric cues. Prior works such as CropFollowRL and CropFollow++ [8, 6] show that vision encodes rich textural and semantic information that can disambiguate row boundaries, weeds, and ground clutter. Future experiments can therefore explore a camera-centered perception stack, benchmarking identical TD–MPC hyperparameters against the present LiDAR condition to isolate modality effects.

In contrast, LiDAR enjoys an intrinsic abstraction advantage: point clouds remain invariant to illumination, color, and crop texture, providing a natural shield against perceptual overfitting when environments change abruptly. Quantifying this trade-off demands controlled cross-sensor studies in which the same latent world model ingests alternate encoders. Such an investigation will illuminate whether hybrid fusion or sensor hand-off policies yield the most reliable field performance.

A good way to improve the algorithm's design is to perform a scaled ablation campaign. Key components scheduled for removal-and-replace tests include (i) the latent ensemble size, (ii) horizon length H, (iii) the auxiliary planning loss $\mathcal{L}_H$ weight, and (iv) the reward regularizers $\lambda_\omega, \lambda_c$. While simple ablations were attempted, comprehensive ablative studies can significantly optimize performance and improve results, particularly in systems with multiple neural networks, as presented.

Moving beyond online-only learning, an *offline-online* hybrid pipeline might be the most promising attempt to be done. This so-called hybrid approach presents sampling each optimizer mini-batch from a mixture of historical replay (β fraction) and the live buffer $(1 - \beta)$. This balanced sampling scheme allows the agent to take advantage of the demonstration data collected before stalling adaptation, a pattern already successful in robot manipulation [13].

The principal motivation for the mixed offline strategy is to limit out-of-training-distribution (OOTD) actions that emerge when exploration drifts far from the replay manifold. Incorporating conservative value uncertainty penalties, in the spirit of FOWM

and prior works [10, 11, 36], is expected to curb catastrophic rollouts and thereby raise the safety envelope required for fully autonomous field trials.

For practical applications, Gazebo may be replaced with alternative simulators offering enhanced robustness for sim-to-real transfer where simulation fidelity is crucial. Furthermore, future work can utilize different TerraSentia robot variations, provided they output (v, w).

Finally, all forthcoming algorithms will include real crop-field validation. Data and pre-trained models will be released under an open-source license to catalyze community replication, with an eye toward establishing a standard benchmark suite for under-canopy navigation across sensor modalities and crop species.

6 Conclusion

This research set out to determine whether a single model–based reinforcement–learning (MBRL) stack could viably replace the classical perception–plus–controller pipeline for under-canopy crop navigation. By embedding environment dynamics, reward structure, and action selection within a shared latent world model, the proposed TD–MPC2 framework succeeds in traversing dense sorghum rows with a marked reduction in lateral deviation and a substantial increase in collision-free completions relative to model-free learning prior works. These findings confirm the central hypothesis: latent-space planning, coupled with learned dynamics, can deliver high-performance control in GPS-denied agricultural settings while reducing the brittle hand-tuning typical of modular controllers.

Beyond the numerical gains, the experiments provide a first glimpse of how MBRL can facilitate few and zero-shot deployment when field conditions change. The LiDAR–only variant demonstrated resilience to illumination and colour variance, whereas preliminary camera-based trials hinted at richer geometric cues for tighter row tracking. Such sensor trade-offs underscore that the present study is an initial waypoint rather than an end-state. Many avenues remain open, including heterogeneous sensor fusion, offline-online hybrid optimization, and hierarchical task planning.

Equally important is the methodological template laid down here. The deliberate decomposition of loss components, the principled evaluation metrics, and the ablation agenda provide a reproducible scaffold for future exploration. These design choices, particularly balanced replay sampling and conservative planning penalties, will guide subsequent efforts to temper out-of-distribution actions and scale the technique to longer horizons and more diverse crops.

While the approach has not yet surpassed finely tuned perception–plus–controller systems, it narrows the gap considerably and does so with far less engineering overhead. The latent-model abstraction offers a promising path toward robots that learn to adapt, rather than being reprogrammed, when confronted with new row geometries, plant morphologies, or terrain irregularities.

Future work will involve integrating contrastive visual pre-training for camera inputs and validating the controller under real-world conditions like wind occlusion and partial canopy collapse. The insights gained will serve as a benchmark and a foundation for developing self-reliant agricultural platforms.

In closing, I wish to express profound gratitude to the collaborators, mentors, and

collegues whose expertise and encouragement have shaped every chapter of this work. Their contributions have transformed an ambitious idea into a concrete framework, and their guidance will continue to inspire the next stage of research seeded by these findings.

References

[1] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator, 2004.

[2] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

[3] Jason DeBruin, Thomas Aref, Sara Tirado Tolosa, Rebecca Hensley, Haley Underwood, Michael McGuire, Chinmay Soman, Grace Nystrom, Emma Parkinson, Catherine Li, Stephen Patrick Moose, and Girish Chowdhary. Breaking the field phenotyping bottleneck in maize with autonomous robots. *Communications Biology*, 8(1), Mar 2025.

[4] Vitor A. H. Higuti, Andres E. B. Velasquez, Daniel Varela Magalhaes, Marcelo Becker, and Girish Chowdhary. Under canopy light detection and ranging-based autonomous navigation. *Journal of Field Robotics*, 36(3):547–567, May 2019.

[5] Francisco Affonso, Felipe Andrade, Gianluca Capezzuto, Mateus V Gasparino, Girish Chowdhary, and Marcelo Becker. Crow: A self-supervised crop row navigation algorithm for agricultural fields. *Journal of Intelligent & Robotic Systems*, 111(1), Feb 2025.

[6] Arun N Sivakumar, Mateus V Gasparino, Michael McGuire, Vitor AH Higuti, M Ugur Akcal, and Girish Chowdhary. Lessons from deploying cropfollow++: Under-canopy agricultural navigation with keypoints. *arXiv preprint arXiv:2404.17718*, 2024.

[7] Andres Eduardo Baquero Velasquez, Vitor Akihiro Hisano Higuti, Mateus Valverde Gasparino, Arun Narenthiran Sivakumar, Marcelo Becker, and Girish Chowdhary. Multi-sensor fusion based robust row following for compact agricultural robots, 2021.

[8] Arun Sivakumar, Ning Wang, Felipe Andrade, G Tommaselli, Mateus Gasparino, Marcelo Becker, and Girish Chowdhary. Cropfollowrl: Learning under-canopy navigation policy with keypoints abstraction. 2024.

[9] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control, 2022.

[10] Yunhai Feng, Nicklas Hansen, Ziyan Xiong, Chandramouli Rajagopalan, and Xiaolong Wang. Finetuning offline world models in the real world, 2023.

[11] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 10 2021.

[12] Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning, 2021.

[13] Haotian Lin, Pengcheng Wang, Jeff Schneider, and Guanya Shi. Td-m(pc)^2: Improving temporal difference mpc through policy constraint, 2025.

[14] Francisco Affonso Pinto, Felipe Andrade G. Tommaselli, Mateus V. Gasparino, and Marcelo Becker. Navigating with finesse: Leveraging neural network-based lidar perception and ilqr control for intelligent agriculture robotics. In *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*, pages 502–507, 2023.

[15] Arun Narenthiran Sivakumar, Sahil Modi, Mateus Valverde Gasparino, Che Ellis, Andres Eduardo Baquero Velasquez, Girish Chowdhary, and Saurabh Gupta. Learned visual navigation for under-canopy agricultural robots. *arXiv preprint arXiv:2107.02792*, 2021.

[16] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[17] Reinis Cimurs, Il Hong Suh, and Jin Han Lee. Goal-driven autonomous exploration through deep reinforcement learning. *IEEE Robotics and Automation Letters*, 7(2):730–737, 2022.

[18] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*, 2018.

[19] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 2018.

[20] Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior, 2022.

[21] Simon Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudik, and John Langford. Provably efficient RL with rich observations via latent state decoding. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1665–1674. PMLR, 09–15 Jun 2019.

[22] Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron Courville, and Alexandre Lacoste. Mastering the unsupervised reinforcement learning benchmark from pixels, 2023.

[23] Muhammad Hafeez Saeed, Hussain Kazmi, and Geert Deconinck. Dyna-pinn: Physics-informed deep dyna-q reinforcement learning for intelligent control of building heating system in low-diversity training data regimes. *Energy and Buildings*, 324:114879, 2024.

[24] Xinyang Gu, Yen-Jen Wang, Xiang Zhu, Chengming Shi, Yanjiang Guo, Yichen Liu, and Jianyu Chen. Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning, 2024.

[25] Chenhao Li, Andreas Krause, and Marco Hutter. Robotic world model: A neural network simulator for robust policy optimization in robotics, 2025.

[26] Shangke Lyu, Xin Lang, Han Zhao, Hongyin Zhang, Pengxiang Ding, and Donglin Wang. Rl2ac: Reinforcement learning-based rapid online adaptive control for legged robot robust locomotion. 07 2024.

[27] Xuemin Hu, Shen Li, Tingyu Huang, Bo Tang, Rouxing Huai, and Long Chen. How simulation helps autonomous driving: A survey of sim2real, digital twins, and parallel intelligence. *IEEE Transactions on Intelligent Vehicles*, 9(1):593–612, 2024.

[28] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 12 2018.

[29] Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[30] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40:1–14, 01 2017.

[31] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[32] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners, 2023.

[33] Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning, 2021.

[34] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[35] Fan-Ming Luo, Tian Xu, Hang Lai, Xiong-Hui Chen, Weinan Zhang, and Yang Yu. A survey on model-based reinforcement learning. *Science China Information Sciences*, 67(2):121101, 2024.

[36] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1702–1712. PMLR, 08–11 Nov 2022.