Extended Abstract

Motivation In autonomous off-road driving, the vehicle dynamics can vary widely depending on the terrain underfoot, especially during aggressive maneuvering. The planner must therefore account for terrain effects to ensure good tracking performance and safety (e.g., to prevent unsafe skidding or rollover). Terrain effects can be difficult to model with traditional physics, motivating learned dynamics models. While offering generally improved accuracy, these learned models may not be globally reliable. Thus, for reliable operation the planner must in turn account for the learned dynamics model uncertainty.

Method Our approach presents two key components: a vision-conditioned learned dynamics model and an uncertainty-aware planner. Using images received from on onboard camera, the dynamics model infers the local terrain and correspondingly modifies the vehicle dynamics. Given a new image, we apply the pre-trained DINO foundation model to assign general-purpose feature descriptors to each image patch, and apply a learned neural network to compress to lower-dimensional terrain latents. For a query state-action pair, we then interpolate to find a terrain latent associated with the query position and pass this terrain latent to a learned neural network for terrain-conditioned tire force prediction. We then incorporate the learned dynamics into an uncertainty-aware model predictive controller. Instead of a single model, we train several models and use the full ensemble for both dynamics prediction and uncertainty quantification. Specifically, we formulate a probabilistic framework for model error, using the ensemble sample covariance, and solve the resulting stochastic model predictive control optimization.

Implementation To learn the vision-conditioned dynamics model, we train using demonstrated trajectories paired with onboard camera images. We train the dynamics model end-to-end, jointly learning the terrain latent and tire force predictor models using single-step prediction error for short trajectory segments. To implement the uncertainty-aware planner, we first solve a certainty-equivalent optimization problem using the ensemble but ignoring model error. Then, using the linearized dynamics we derive an optimal linear tracking controller to model closed-loop error propagation. Lastly, using the ensemble sample covariance and this tracking controller we solve an uncertainty-aware optimization problem, featuring an additional uncertainty term in the cost.

Results We test our method using a visually realistic simulator fitted to a real room using Gaussian splatting. This room features visually distinct tiles which we artificially assign different terrain properties. We evaluate our method's performance using reference tracking cost and the fraction of trajectories diverging from the path reference. We compare against a terrain-agnostic baseline which assumes a single terrain across space as well as an ensemble baseline which uses the ensemble average for the dynamics model but ignores model uncertainty. Our results show that we improve significantly in tracking cost and divergence fraction compared to the ensemble baseline. While our method achieves higher average tracking cost than the terrain-agnostic baseline (due to some instances of divergence), our median performance is better.

Discussion In addition to quantitative analysis, we present example results visualizing the learned terrain latents using our model. These latents distinguish the different terrains, qualitatively suggesting the validity of our vision-conditioned dynamics model. Additionally, we compare the trace of the ensemble covariance for our uncertainty-aware planner and the uncertainty-agnostic ensemble baseline. We observe that the trace is reduced using our method, suggesting that the uncertainty-aware planner prevents the vehicle from reaching regions where the learned ensemble dynamics have high uncertainty.

Conclusion We developed a prototype approach for off-road autonomous driving. Using a vision-conditioned learned dynamics model we infer terrain effects, and couple this with an uncertainty-aware planner to improve reliability by accounting for model error. We test our approach using a visually realistic simulator and demonstrate improved performance compared to terrain-agnostic and uncertainty-agnostic baselines.

Uncertainty-Aware Planning for Off-road Autonomous Driving with Vision-Conditioned Learned Dynamics

Aaron Feldman
Department of Aeronautics and Astronautics
Stanford University
aofeldma@stanford.edu

Abstract

In this project, we develop a prototype approach for off-road autonomous driving using a learned dynamics model to account for terrain. To identify terrain effects, we condition the dynamics model on images received from an onboard camera. Our dynamics model reasons locally about the terrain associated with different image patches, so our planner can anticipate future dynamics changes ahead of the vehicle. Since we train the dynamics model using demonstrated trajectories, it may not be globally reliable. Therefore, we explicitly account for model error, formulating an uncertainty-aware model predictive control strategy for action planning. To quantify model uncertainty, we train an ensemble of dynamics models. Using the ensemble, we derive a probabilistic framework for model error and solve a stochastic model predictive control problem using the ensemble covariance for uncertainty quantification. We evaluate our approach in visually realistic simulation and improve tracking performance compared to terrain-agnostic and uncertainty-agnostic baselines.

1 Introduction

For effective and safe autonomous off-road driving, the robot must be able to reason about the effects of terrain on the vehicle dynamics and account for this in its planning Han et al. (2024). For instance, the robot planner must account for increased slip over a sandy surface, slow down before an ice patch to prevent skid, or avoid bumps which could cause rollover. However, accurate physical modeling of the dynamic effects induced by terrain can be challenging, especially when the vehicle drives aggressively or near performance limits Djeumou et al. (2023); Kabzan et al. (2019). The vehicle may encounter a wide and continuously varying set of terrains in the real-world and the associated dynamic effects may be complex. Therefore, rather than predefining physical models for the terrain, in this project we propose to learn a terrain-conditioned dynamics model from trajectory data. Using this model, we can predict vehicle state evolution conditioned on the terrain underfoot. To enable effective planning, this dynamics model should provide insight not only about the immediate terrain effects but also foresight into upcoming terrain effects ahead of the vehicle. For instance, this foresight is necessary for the planner to anticipate the effects of an upcoming ice patch and slow down accordingly. To achieve this foresight with our dynamics model, we propose to condition the learned dynamics model using image input. Using a forward-facing onboard camera capturing images of the nearby terrain, our dynamics model learns to associate visual features of the image with terrain effects, assigning a terrain latent to each pixel patch in the image. By then interpolating these latents based on a queried position, we can infer the associated terrain underfoot and learn to predict the associated tire forces. Notably, we learn the terrain-conditioned dynamics model in a single training phase, jointly learning the terrain latent model and tire force predictor model. By doing so, we can learn to extract visual information which specifically pertains to dynamics prediction i.e., terrain effects. Additionally, while our approach relies on trajectory data to learn the vision-conditioned dynamics model, these trajectories do not need to be expert demonstrations. Moreover, because we separately learn a dynamics model, it can then be used for a wide variety of planning tasks downstream and possibly new environments with similar terrains.

While this model-based approach offers flexibility, a key challenge is that the learned dynamics model is imperfect and not reliable across the full state space. Rather, the model is a function of the trajectory training data, which may cover a limited subset of the possible state, action, and terrain combinations. For instance, if the trajectory demonstrations come from teleoperation they may not feature many instances of dramatic acceleration or high speed. As we show in our results, it is critical to account for the learned dynamics model inaccuracy in planning. Without doing so, the planner can venture into regimes of poor dynamics prediction, resulting in divergence from the intended motion. Therefore, to complement our vision-conditioned learned dynamics model, we develop an uncertainty-aware planner. We adopt a model predictive control (MPC) framework, where we repeatedly plan actions for a short horizon into the future by online optimizing a tracking cost using the dynamics model. To account for model uncertainty, instead of using a single model, we train an ensemble of learned dynamics models. Then, during planning we use the ensemble sample covariance to estimate model uncertainty and optimize the planning objective in expectation, accounting for model uncertainty in the cost.

We test our approach in a visually realistic simulator, using a Gaussian splat Ye et al. (2024) to render realistic camera views of the terrain. We show that our vision-conditioned dynamics model distinguishes the different terrain types and that our planner improves tracking performance compared to a terrain-agnostic baseline which assumes a single, global terrain. Furthermore, we demonstrate that accounting for model uncertainty is critical during planning, as an ablation which plans using the ensemble average without an additional uncertainty term performs poorly and frequently diverges.

In summary, the key contributions of this work are:

1. From demonstrated trajectories, we learn a vision-conditioned dynamics model to account for terrain effects during autonomous off-road driving.
2. We incorporate this learned dynamics model into an uncertainty-aware MPC planner which can anticipate future terrain changes and account for model uncertainty using a model ensemble.
3. In visually realistic simulation, we show that our approach improves performance over a terrain-agnostic baseline as well as using an ensemble model but without additional uncertainty quantification.

2 Related Work

Prior work has shown that visual input can successfully be used to inform planning over varied terrain. All these works aim to associate visual features with terrain effects, but vary in how they describe the terrain effects. Some predict an associated cost Castro et al. (2023) or traversability Frey et al. (2023); Mattamala et al. (2024), others explicit/interpretable physics parameters Chen et al. (2024); Margolis et al. (2023), or learned terrain latent parameters Gibson et al. (2024); Nagabandi et al. (2018). Of this category, most similar to ours is Gibson et al. (2024) which also leverages the pre-trained DINO model Oquab et al. (2024) to obtain general-purpose image features. The DINO latents are then registered to a voxel location and input to a learned dynamics model.

An additional line of work focuses on meta-learning and online adaptation for learned dynamics models. Meta-learning approaches Harrison et al. (2018); Banerjee et al. (2020); Richards et al. (2021, 2022); O'Connell et al. (2022); Lapandić et al. (2024) seek to learn a base dynamics model offline which can be quickly adapted online as state transitions are observed. These approaches can also be combined with vision to improve the base dynamics model Lupu et al. (2024); Levy et al. (2025). Adaptation can also be framed as learning a dynamics-conditioning latent from a buffer of the recent state history Djeumou et al. (2024); Kumar et al. (2021). While valuable, meta-learning and online adaptation are complementary to our work that could serve as valuable extensions. Instead of reasoning in hindsight from recent transitions or only about the immediate terrain, we focus on using vision-conditioning so our planner can anticipate future dynamic changes. Furthermore, we incorporate model uncertainty directly into our planner, anticipating model error in the future planning horizon rather than focusing on adapting the model online based on observed error.

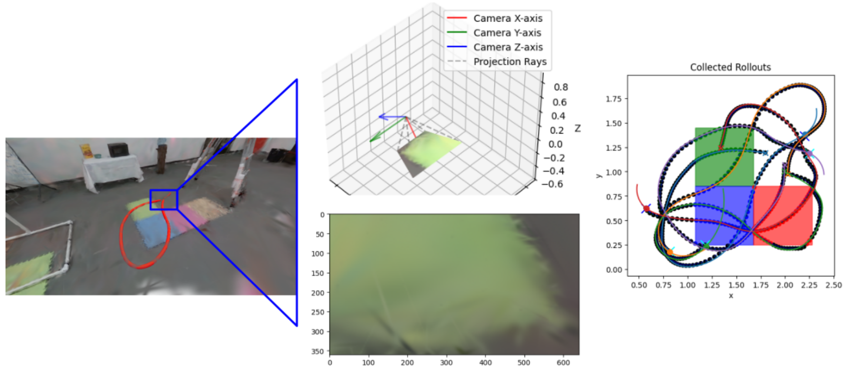

Figure 1: Visualizing Trajectory Training Data

Lastly, there have been other works using ensembles of learned models for uncertainty-aware planning/control. Training an ensemble of neural network models and studying their disagreement has been proposed as a general approach for neural network uncertainty quantification Lakshminarayanan et al. (2017). In model-based reinforcement learning, an ensemble is often used to account for error in the transition dynamics model Buckman et al. (2019); Kurutach et al. (2018). The most similar work to ours are Chua et al. (2018); Dyro et al. (2021) which fit an ensemble of neural network dynamics models and then uses them for online action planning. They propose to sample transitions from each model in the ensemble to get a set of possible rollouts (i.e., particles) when optimizing for an action sequence. In contrast, our proposed approach propagates the dynamics using the ensemble average, but accounts for stochasticity by appropriately modifying the optimization objective based on the ensemble sample covariance. Yu et al. (2020) explores a similar idea of penalizing the reward based on ensemble uncertainty for offline reinforcement learning. In contrast with Yu et al. (2020), our approach shows how to penalize based on a principled probabilistic framework for model error using the ensemble, rather than a hyperparameter-weighted heuristic penalty.

3 Method

Our method consists of two key components: the vision-conditioned learned dynamics model and the uncertainty-aware planner using it.

3.1 Vision-Conditioned Learned Dynamics Model

To learn the vision-conditioned dynamics model we use trajectory data paired with video from the onboard camera. At each trajectory time, we record the vehicle state X_t, the action taken U_t, and current image I_t of the terrain ahead of the vehicle. In our case, we generate trajectories using an expert MPC, which knows the ground-truth physics across terrains in the simulation, and tracks randomly generated reference paths. However, the trajectory data could come from human operation, or via a simplified feedback controller. Since the data is used only for dynamics learning, it need not be optimal. Fig. 1 shows an example trajectory generated in this way, an associated image from one timestep rendered in the Gaussian splat simulator and its projection onto the ground ahead of the vehicle, and several trajectories with associated references.

We model the vehicle using a bicycle model Hoffmann et al. (2007) where state $X = (x, y, \psi, v_x^B, v_y^B, \omega)$ consists of the global position $p = (x, y)$ and heading ψ, longitudinal and lateral speeds in body frame $v = (v_x^B, v_y^B)$, and angular rate ω. The commanded action $U = (F_c, \delta)$ where F_c is a commanded thrust force and δ is the forward steering angle. The vehicle dynamics are given by

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v}_x^B \\ \dot{v}_y^B \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} v_x^B \cos(\psi) - v_y^B \sin(\psi) \\ v_x^B \sin(\psi) + v_y^B \cos(\psi) \\ \omega \\ 1/m(F_x - F_{yf} \sin(\delta) + m v_y^B \omega) \\ 1/m(F_{yr} + F_{yf} \cos(\delta) - m v_x^B \omega) \\ 1/I_z(F_{yf} a \cos(\delta) - F_{yr} b) \end{pmatrix} \tag{1}$$

where a, b, m, I_z are the vehicle forward/rear axle distances, mass, and moment of inertia. The tire forces (F_x, F_{yr}, F_{yf}) are a longitudinal force, and two lateral forces at the rear/forward axles. These tire forces depend on the terrain beneath the vehicle, and have previously been modeled using physics (e.g., using the slip angle) or learned models Kabzan et al. (2019); Djeumou et al. (2023). Our goal will be to learn these tire forces using an image of the terrain, as described below.

Our vision-conditioned dynamics model operates in several stages. Given a conditioning image I of the terrain and a query state X_t and associated action U_t, our model $\hat{f}$ aims to predict the resulting next state $X_{t+1} \approx \hat{f}(X_t, U_t; I)$. The overall model architecture is summarized in Fig. **??**.

When receiving a new image I, we first apply the pre-trained DINO model Oquab et al. (2024) to it, obtaining a set $z_1, ..., z_D$ of general-purpose features $z_i \in \mathbb{R}^{384}$, describing each 14×14 pixel patch in the image. We then compress these general-purpose features to lower-dimensional terrain latents $\theta_1, ..., \theta_D$ using a neural network learned module $h(z_i) = \theta_i \in \mathbb{R}^3$ applied identically and separately to each DINO feature. The hope is for h to extract from the DINO features only the relevant visual information pertinent for terrain classification and dynamics prediction. These resulting terrain latents $\{\theta_i\}_{i=1}^D$ can be used repeatedly until a new image is obtained. By using the camera parameters, we can project from image coordinates to physical coordinates in space (see the center portion of Fig. 1) relying on a flat ground assumption or depth image. Thus, we can obtain 2D positions on the ground $\{p_i\}_{i=1}^D$ for each pixel patch, associated with the terrain latents $\{\theta_i\}_{i=1}^D$.

Given a query state X and action U, we first perform interpolation to determine a terrain latent θ describing the terrain at the query position. Given the query position p, we perform radial basis function interpolation

$$\theta(p) = \sum_{i=1}^{D} w_i \theta_i, w_i = \exp(-\gamma ||p - p_i||_2^2) / \sum_{j=1}^{D} \exp(-\gamma ||p - p_j||_2^2). \tag{2}$$

We then use θ as an additional input for a learned neural network tire force module

$$(F_x, F_{yr}, F_{yf}) = g(v_x^B, v_y^B, \omega, F_c, \delta, \theta). \tag{3}$$

After obtaining the tire forces, we apply the bicycle model Eq. 1 with Euler discretization to predict the next resulting state. Notably, our learned dynamics $\hat{f}$ do not depend on global coordinates, so our approach can generalize to novel environments and tasks (featuring similar terrains as seen in training).

We train the dynamics model end-to-end, jointly learning the terrain latent model h and tire force predictor g. To do so, we randomly extract short trajectory segments from the data $(I_0, X_0, U_0, X_1, U_1, ..., X_N)$ and compute the single-step prediction error when conditioning on the starting image I_0 for all times:

$$Loss = \sum_{k=0}^{N-1} ||X_{k+1} - \hat{f}(X_k, U_k)||_2^2 \tag{4}$$

We choose N based on our MPC planner's horizon. By reusing I_0 across the horizon, the learned dynamics model is trained to use different parts of the image depending on the query state X_k distance to X_0 (e.g., the bottom of the image is used earlier in the horizon and upper portions towards the end).

3.2 Uncertainty-Aware Planner

We use the fitted vision-conditioned dynamics model for MPC planning to track a given reference. MPC repeatedly plans for a short horizon of N actions, using the dynamics model $\hat{f}$ to predict future

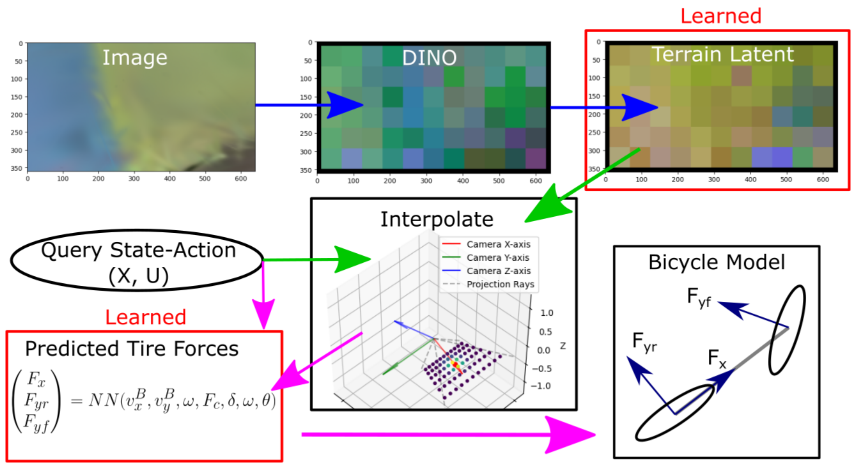

Figure 2: Vision-Conditioned Dynamics Model Architecture

states, and optimizes a tracking objective to follow a reference path X_g:

$$\begin{aligned} \min_{U_{0:N-1}, X_{1:N}} \quad J = \sum_{k=0}^{N} ||X_k - X_g^k||_Q^2 + \sum_{k=0}^{N-1} ||U_k - U_{k-1}||_{R_\Delta}^2 \\ \text{s.t. } X_{k+1} = \hat{f}(X_k, U_k; I_0), \quad \forall k = 0 : N-1, \\ F_u U_k \leq g_u, \quad \forall k = 0 : N-1 \end{aligned} \tag{5}$$

Here $Q \succ 0$ is used to penalize state deviation from the reference states, $R_\Delta \succ 0$ is used to encourage temporally smooth action sequences [1]. F_u, g_u describe control constraints, and we condition $\hat{f}$ on the latest image I_0 throughout the horizon. After solving for an open-loop plan $U_{0:N-1}$, $X_{1:N}$, MPC executes U_0 before re-solving at the next time.

Directly solving Eq. 5 with a single model performs quite poorly, as it neglects model error/uncertainty in the learned dynamics $\hat{f}$. To improve performance, we propose training an ensemble of several dynamics models $\hat{f}_1, ..., \hat{f}_M$, using them for both improved dynamics prediction and model uncertainty estimation. Our approach adopts a probabilistic model of the dynamics using the ensemble and then solves the resulting stochastic MPC problem.

Each neural network is trained on the same data but with random initialization and stochastic descent, therefore for given query (X, U) the errors ϵ_i are independent, identically distributed random variables

$$f(X, U) - \hat{f}_i(X, U) = \epsilon_i, \mathbb{E}[\epsilon_i] = 0, \text{Cov}[\epsilon_i] = \Sigma(X, U) \tag{6}$$

where $f(X, U)$ is the fixed but unknown true dynamics. Our only assumption/approximation is that the error is zero-mean i.e., if we were to train many neural network models, on average, they would predict the correct value $f(X, U)$. From Eq. 6, the ensemble average dynamics $\overline{f} = 1/M \sum_{i=1}^{M} \hat{f}_i$ have random error $\overline{\epsilon} = 1/M \sum_{i=1}^{M} \epsilon_i$ yielding

$$f(X, U) = \overline{f}(X, U) + \overline{\epsilon}, \mathbb{E}[\overline{\epsilon}] = 0, Cov(\overline{\epsilon}) = \Sigma(X, U)/M. \tag{7}$$

Since the error covariance is unknown, we approximate it by the ensemble sample covariance

$$\Sigma(X, U) \approx \hat{\Sigma}(X, U) = \frac{1}{M-1} \sum_{i=1}^{M} (\hat{f}_i(X, U) - \overline{f}(X, U))(\hat{f}_i(X, U) - \overline{f}(X, U))^T. \tag{8}$$

Using Eq. 7 we can describe the ensemble error after a single time-step $X_{k+1} = f(X_k, U_k) = \overline{f}(X_k, U_k) + \overline{\epsilon}_k$. However, we need to consider how error propagates over the N-step horizon. We achieve this by using a linear approximation of the dynamics and a linear tracking controller as a

[1] For U_{-1} in Eq. 5 we use the executed control action from the previous timestep.

surrogate for closed-loop execution of MPC. We first solve for an initial plan $\{\tilde{X}_k\}_{k=0}^N, \{\tilde{U}_k\}_{k=0}^N$ solving the optimization in Eq. 5 with the ensemble dynamics and ignoring model error. Using this initial plan, we linearize the ensemble dynamics $\overline{f}$ about these states and actions to approximate

$$X_{k+1} \approx A_k X_k + B_k U_k + C_k + \overline{\epsilon}_k. \tag{9}$$

Ideally, we would like to understand how error propagates under closed-loop execution of MPC, which re-solves at each time and can thus respond to new errors. As an implementable surrogate model, we instead use the linearized dynamics $\{(A_k, B_k)\}_{k=0}^{N-1}$ to derive an optimal linear feedback controller $\{K_k\}_{k=0}^{N-1}$ using the backwards Riccati equation (see the Appendix for details). Thus, for the purposes of optimization, we approximate future MPC's response to error $e_k = X_k - \overline{X}_k$ via

$$U_k = \overline{U}_k + K_k(X_k - \overline{X}_k) \tag{10}$$

where $\overline{U}_k$ is the nominal/planned action and $\overline{X}_k$ is the nominal/planned state using the ensemble dynamics. From a practical perspective, introducing the tracking controller is necessary to ensure that the error dynamics are stable, preventing cost divergence. The nominal states evolve under the ensemble dynamics

$$\overline{X}_{k+1} = \overline{f}(\overline{X}_k, \overline{U}_k) \approx A_k \overline{X}_k + B_k \overline{U}_k + C_k. \tag{11}$$

Thus, using Eq. 9, Eq. 11, and Eq. 10 error dynamics are approximately given by

$$e_{k+1} = (A_k - B_k K_k) e_k + \overline{\epsilon}_k. \tag{12}$$

This can be written in batch form to express $E = S_\epsilon \mathcal{E}$ where the matrix S_ϵ is lower diagonal and uses the tracked dynamics $A_k + B_k K_k$ for error propagation and $\mathcal{E}$ is column stack of $\{\overline{\epsilon}_k\}_{k=0}^{N-1}$. In batch, we therefore write $X = \overline{X} + S_\epsilon \mathcal{E}$ where $\overline{X}$ is the deterministic, nominal states over time. Similarly, the tracking feedback law yields $U = \overline{U} + \overline{K} S_\epsilon^c W$ where $\overline{K} = \text{blockdiag}(K_0, K_1, ..., K_{N-1})$ and S_ϵ^c is the same as S_ϵ except that the last block row is clipped (since e_N is irrelevant).

For tractability, we assume $E[\mathcal{E}] = 0$, $Cov[\mathcal{E}] = \overline{\Sigma} = \text{blockdiag}(\hat{\Sigma}_0/M, \hat{\Sigma}_1/M, \ldots, \hat{\Sigma}_{N-1}/M)$. In other words, we additionally assume that the ensemble prediction error is uncorrelated across time. We use the ensemble covariance to estimate the one-step prediction error covariance $\hat{\Sigma}_i/M$, queried at $\overline{X}_k, \overline{U}_k$.

Under our probabilistic framework, the cost J is stochastic, so for optimization, we minimize the expected cost

$$\mathbb{E}[J] = \mathbb{E}[(X - X_g)^T \overline{Q}(X - X_g) + (U - U_-)^T \overline{R}_\Delta (U - U_-)] \tag{13}$$

where $U_- = (U_{-1}, U_0 ..., U_{N-2})$, $\overline{Q} = \text{blockdiag}(Q, ..., Q)$, and $\overline{R}_\Delta = \text{blockdiag}(R_\Delta, ..., R_\Delta)$.

Using the trace trick $\mathbb{E}[z^T z] = \text{tr}(\mathbb{E}[zz^T])$ and canceling cross-terms assuming $E[\mathcal{E}] = 0$ yields

$$E[J] = (\overline{X} - X_g)^T \overline{Q}(\overline{X} - X_g) + \overline{U}^T \overline{RU} + \text{tr}(\overline{\Sigma} D) \tag{14}$$

where $D = D_X + D_{\Delta U}$ is constant with respect to the decision variables and describes how the error propagates and effects the cost over time. Here $D_X = S_\epsilon^T \overline{Q} S_\epsilon$ and $D_{\Delta U} = (M\overline{K}S_\epsilon^c)^T \overline{R}_\Delta (M\overline{K}S_\epsilon^c)$ where M is a first-order block difference matrix.

The first term in Eq. 14, $(\overline{X} - X_g)^T \overline{Q}(\overline{X} - X_g) + \overline{U}^T \overline{RU}$ is simply the objective using the nominal ensemble dynamics without the model error. The second term in Eq. 14, $\text{tr}(\overline{\Sigma} D) = \sum_{k=0}^{N-1} \text{tr}(\frac{1}{M}\hat{\Sigma}_k D_{kk})$ (as $\overline{\Sigma}$ is a block-diagonal) and D_{kk} refers to the k-th diagonal block of D.

The resulting uncertainty-aware optimization problem is

$$\begin{aligned} \min_{\overline{U}_{0:N-1}, \overline{X}_{1:N}} \quad \mathbb{E}[J] = \sum_{k=0}^{N} ||\overline{X}_k - X_g^k||_Q^2 + \sum_{k=0}^{N-1} ||\overline{U}_k - \overline{U}_{k-1}||_{R_\Delta}^2 + \sum_{k=0}^{N-1} \text{tr}(\frac{1}{M}\hat{\Sigma}(\overline{X}_k, \overline{U}_k) D_{kk}) \\ \text{s.t. } \overline{X}_{k+1} = \overline{f}(\overline{X}_k, \overline{U}_k; I_0), \quad \forall k = 0 : N-1, \\ F_u \overline{U}_k \leq g_u, \quad \forall k = 0 : N-1 \end{aligned} \tag{15}$$

In summary, our uncertainty-aware MPC planner performs the following steps at each time:

1. Solve the certainty-equivalent optimization problem Eq. 5 using the ensemble dynamics to obtain $\{\tilde{X}_k\}_{k=0}^N$, $\{\tilde{U}_k\}_{k=0}^{N-1}$ and linearize the dynamics about this guess, obtaining $\{(A_k, B_k, c_k)\}_{k=0}^{N_1}$.
2. Using the linearized dynamics, derive a linear feedback controller with gains $\{K_k\}_{k=0}^{N-1}$ via backwards Riccati recursion.
3. Solve the uncertainty-aware optimization problem Eq. 15 using $\{K_k\}_{k=0}^{N-1}$ for closed-loop error propagation and ensemble covariances $\{\hat{\Sigma}_k\}_{k=0}^{N-1}$ for uncertainty quantification.

4 Experimental Setup

To test our method, we develop a visually realistic simulator by using Gaussian splatting to render images from a real room. On the floor, there are three tiles (red, green, and blue) and we artificially assign to each of these tiles different terrain properties and use a single terrain for the remainder of the room. Specifically, the true terrain dynamics, which are unknown to the model and must be learned from vision, are that the lateral tire forces are proportional to the slip angle $F_{yr/f} = C_y \alpha_{r/f}$ where $\alpha_r = \tan^{-1}(\frac{v_y^B - b\omega}{v_x^B})$ and $\alpha_f = \tan^{-1}(\frac{v_y^B + a\omega}{v_x^B}) - \delta$. Intuitively, smaller in magnitude C_y values correspond to more slippery surfaces. We set $C_y = -1$ for the red terrain, $C_y = -2$ for the green terrain, $C_y = -5$ for the blue terrain, and the remaining background terrain has $C_y = -10$. The true model for the longitudinal force, is terrain independent (but is also learned) is $F_x = F_c - F_0$ where $F_0 = 0.1$ models rolling resistance.

To train the vision-conditioned dynamics ensemble, we fit $M = 5$ models, each to the same training data of 400 trajectories, each consisting of 100 timesteps with discretization of $\Delta t = 0.05$. The randomly generated reference trajectories (tracked by the expert MPC for training data) were chosen to traverse the tiled portions of the floor, and some examples are shown in Fig. 1. The appendix contains more details of the dynamics architecture hyperparameters.

We use a sampling-based solver for model predictive control, drawn from Howell et al. (2022), which randomly generates many perturbations to the previous timesteps action sequence, and selects the one achieving minimum cost. Despite its simplicity, it works surprisingly well, and a key insight of the approach if that the perturbations are generated using randomly set splines, ensuring that the perturbed action sequences will still be temporally smooth. We plan with a horizon of $N = 10$ and use $Q = \text{diag}(1, 1, 0, 0, 0, 0)$ for position-only penalization and $R_\Delta = \text{diag}(0.05, 0.05)$.

5 Results

We evaluate our method by considering performance on a random set of 50 test references, generated in the same fashion as the training data. As performance metrics we consider the tracking cost over the full resulting closed-loop trajectory, defined as in Eq. 5 with the same Q and R_Δ but for the full trajectory duration. Since a key concern is that without accounting for model error, the planner might diverge, we define a divergence metric. We declare a trajectory to have diverged from the reference when the final state is more than 0.5 [m] from the associated goal state. As summary statistic over the test runs, we consider the median total cost along with interquartile range IQR (i.e., the difference between the 75th and 25th percentiles) as a measure of spread. We use median and IQR as measures which will be less sensitive to outliers i.e., instances of divergence. Instead, we separately report the fraction of divergence. For completeness, we also compute the average cost and associated 2σ confidence bounds, although this measure is more sensitive to divergence outliers.

We compare our method against several alternatives. To study our suboptimality, we compare against an "oracle" MPC which uses the true dynamics unknown to our method. We also consider a terrain-agnostic "default" baseline which assumes the relationship $F_y = C_y \alpha$ but always uses a $C_y = -4.5$, averaging across the terrains. By comparing against this method, we can observe the potential gain of reasoning about terrain effects. Lastly, we consider an "ensemble" ablation which solves the certainty-equivalent optimization Eq. 5 still using the ensemble average dynamics, but dropping the ensemble uncertainty term found in Eq. 15. By comparing against this method, we can assess the benefit our uncertainty-aware planner.

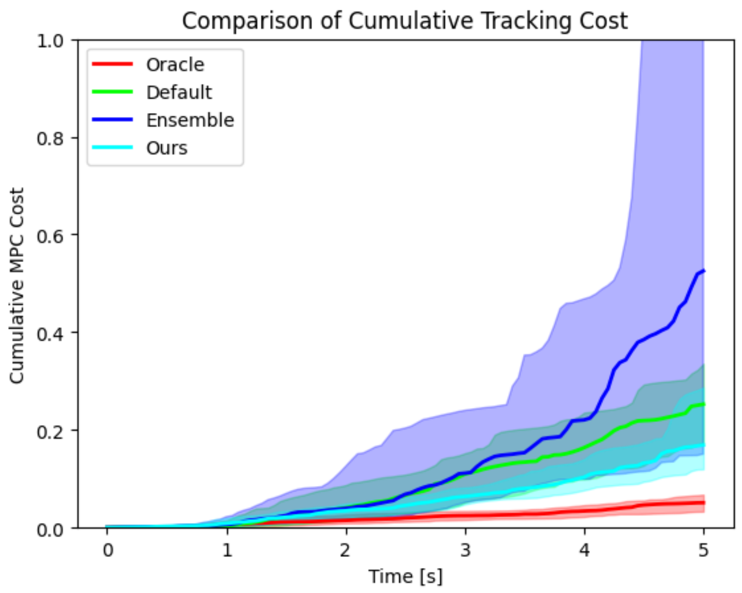

Figure 3: Cumulative Tracking Cost over Time

Table 1: Performance Comparison

Method	Median Cost	IQR	Average Cost, 2σ CI	Divergence Fraction
Oracle	0.051	0.034	0.065, (0.049, 0.08)	0.0
Default	0.253	0.162	0.288, (0.232, 0.344)	0.0
Ensemble	0.525	1.947	1.72, (1.000, 2.441)	0.12
Ours	0.169	0.167	0.412, (0.111, 0.713)	0.02

5.1 Quantitative Evaluation

In Figure 3 we compare the cumulative tracking cost across the trajectory time for each of the methods (Oracle, Default, Ensemble, and Ours). Specifically, we plot the median tracking cost up to the given time and shade the 25th to 75th percentile region. We observe that oracle unsurprisingly performs best, followed by our method, then the terrain-agnostic default, and lastly the ensemble ablation with no uncertainty term. We also note that the interquartile range grows dramatically for the ensemble approach towards the end, reflecting potential divergence.

We present summary statistics for the final time of these runs in Table 1. After oracle, our method achieves the lowest final median cost followed by default and then ensemble. The interquartile range for oracle is lowest, with comparable ranges fo default and our method, and a higher interquartile range for the ensemble. Intuitively, the interquartile range results indicate the oracle performs most consistently and ensemble least consistently. This is reflected also looking at the divergence fraction: oracle and default never diverge, our method does 2% of the time, while ensemble diverges 12%. This finding supports the important of uncertainty-aware planning to prevent divergence, although it is still imperfect. Although default has a higher median final cost that our method, it outperforms in terms of the final average cost. This contrast can reflect the presence of some poorly performing outliers when using our method. Further improving our method to prevent the remaining cases of divergence is an important next step, and could even potentially be accomplished by simply adding more models to the ensemble.

5.2 Qualitative Analysis

We also performed qualitative analysis using single example trajectory runs to better understand the effectiveness of the learned dynamics model and the uncertainty-aware planner.

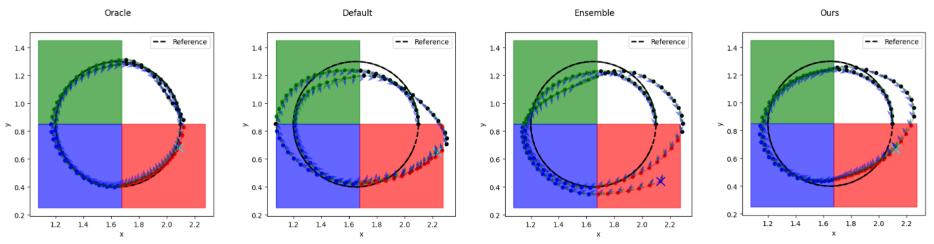

Figure 4: Performance on Challenging Circle Reference

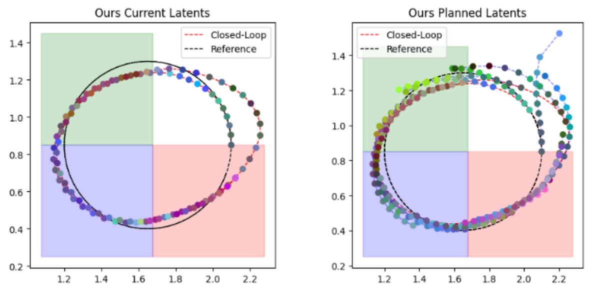

Figure 5: Current and Planned Terrain Latents

Figure 4 shows an example trajectory for each of the methods. We chose this example to follow a circular reference of radius 0.45 [m] at a reference speed of 1.1 [m/s]. This was chosen as a particularly challenging example due to its fast speed at a relatively tight radius. Qualitatively, while the oracle tightly follows the circle, we observe that the terrain-agnostic default slips out on the green and red terrains which have lower magnitude C_y. The ensemble does so as well and begins to diverge. In contrast, our method performs relatively well, slipping out only on the red terrain and not as extensively.

For this same example, we also visualized with RGB coloring the interpolated terrain latents $\theta \in \mathbb{R}^3$ (averaged across the ensemble models) for our method. [2] Figure 5 shows the resulting latents. On the left, we show only the latent at the current state over time. On the right, we visualize at different times the N latents for the full planned MPC horizon. We observe that there are indeed roughly four colors in the left plot showing the current latents, reflecting that the vision-conditioned dynamics model is capable of distinguishing the terrains. The right plot showing the planned latents is more ambiguous but still shows some colors shifts across the different terrains. Notably, we can also see some instances where early in the horizon the latent is one color but then shifts to another as the terrain shifts. For instance, in the upper portion of the plot we can see transitions from blue to green. Although not clearly defined, this loose finding can suggest the capability of our vision-conditioned dynamics model to anticipate terrain shifts later in the planning horizon.

Lastly, to qualitatively understand the impact of the uncertainty term, we visualized in Figure 6 the trace of the ensemble covariance when queried at $(X_0, \overline{U}_0)$, comparing the ensemble approach with our approach featuring the additional uncertainty term. We observe that our approach tends to reduce the ensemble covariance trace relative to the baseline with no uncertainty term. In particular, our uncertainty-aware planner reduces spikes in the trace evident in the baseline. These findings suggest that the uncertainty term encourages the planner to remain in regions where the ensemble models agree, which may help prevent divergence.

6 Discussion

Our approach demonstrated a proof-of-concept for uncertainty-aware vehicle planning using a vision-conditioned learned dynamics model. However, there are several key directions to continue this work.

[2]For better display, we used PCA to align RGB with principle components.

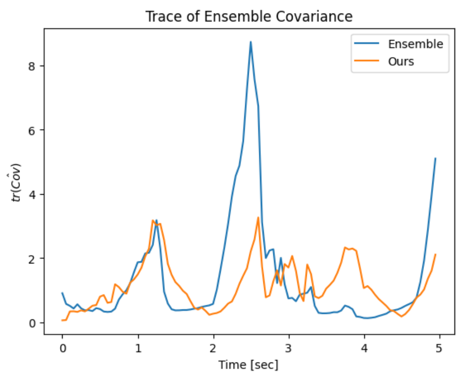

Figure 6: Comparing Ensemble Covariance Trace with and without Uncertainty Term

Firstly, testing in hardware using an RC or full-scale car and in varied off-road environments would help determine the real-world viability of the proposed approach. Secondly, it would be valuable to compare our approach against alternative dynamics models using history-conditioning and adaptation. These approaches could also be incorporated into our vision-conditioned model. Thirdly, it would be valuable to compare against particle-based approaches for uncertainty-aware MPC planning.

From the theoretical perspective, it would be valuable to empirically justify/assess the probabilistic framework for using the ensemble (e.g., the zero-mean error assumption and accuracy of the ensemble sample covariance). This probabilistic framework could also be extended for other uses. Specifically, it could be used to explicitly constrain the planner to avoid traversing unknown/anomalous terrains. This could be achieved by adding an optimization constraint bounding the trace of the ensemble covariance (i.e., the expected squared single-step dynamics error should remain sufficiently small). Additionally, the same error propagation approach currently used for uncertainty-aware planning could also be used to enforce chance constraints on the state e.g., to avoid obstacles or occluded regions while accounting for the model error.

While our approach significantly improved on the ensemble baseline, it can still diverge. To resolve this issue, we could try adding more models to the ensemble or improve the quality of each model. One method to improve model quality could be to train the models using a closed-loop tracking cost with linear feedback (as currently done for error propagation) rather than mean-squared error. Doing so could provide a loss function better aligned with the downstream planning use case for the model and thereby potentially achieve improved performance.

7 Conclusion

In this work, we developed a prototype approach for terrain-dependent modeling and planning for an autonomous off-road vehicle. We learned a vision-conditioned dynamics model to infer terrain effects on the vehicle dynamics using an onboard camera. We then incorporated this learned model into an uncertainty-aware planner using an ensemble of models to account for model error in the planning procedure. We tested our approach in visually realistic simulation and demonstrated improved performance relative to a terrain-agnostic baseline as well as a planning method which uses the ensemble but neglects model uncertainty. While we developed our approach for off-road autonomous driving, it could be adapted for other dynamics learning problems. For instance, vision could be used to inform model-based planning for manipulation or to reason about currents and wind for autonomous shipping.

Changes from Proposal Given limited time, we left for future work the goals of comparing against a particle-based planner and implementing a tracking-based loss for model training.

References

S. Banerjee, J. Harrison, P. M. Furlong, and M. Pavone. 2020. Adaptive Meta-Learning for Identification of Rover-Terrain Dynamics. arXiv:2009.10191 [cs.RO] `https://arxiv.org/abs/2009.10191`

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. 2019. Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion. arXiv:1807.01675 [cs.LG] `https://arxiv.org/abs/1807.01675`

Mateo Guaman Castro, Samuel Triest, Wenshan Wang, Jason M. Gregory, Felix Sanchez, John G. Rogers III, and Sebastian Scherer. 2023. How Does It Feel? Self-Supervised Costmap Learning for Off-Road Vehicle Traversability. arXiv:2209.10788 [cs.RO] `https://arxiv.org/abs/2209.10788`

Jiaqi Chen, Jonas Frey, Ruyi Zhou, Takahiro Miki, Georg Martius, and Marco Hutter. 2024. Identifying Terrain Physical Parameters from Vision – Towards Physical-Parameter-Aware Locomotion and Navigation. arXiv:2408.16567 [cs.RO] `https://arxiv.org/abs/2408.16567`

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. arXiv:1805.12114 [cs.LG] `https://arxiv.org/abs/1805.12114`

Franck Djeumou, Jonathan Y. M. Goh, Ufuk Topcu, and Avinash Balachandran. 2023. Autonomous Drifting with 3 Minutes of Data via Learned Tire Models. arXiv:2306.06330 [eess.SY] `https://arxiv.org/abs/2306.06330`

Franck Djeumou, Thomas Jonathan Lew, NAN DING, Michael Thompson, Makoto Suminaka, Marcus Greiff, and John Subosits. 2024. One Model to Drift Them All: Physics-Informed Conditional Diffusion Model for Driving at the Limits. In *8th Annual Conference on Robot Learning*. `https://openreview.net/forum?id=0gDbaEtVrd`

Robert Dyro, James Harrison, Apoorva Sharma, and Marco Pavone. 2021. Particle MPC for Uncertain and Learning-Based Control. arXiv:2104.02213 [eess.SY] `https://arxiv.org/abs/2104.02213`

Jonas Frey, Matias Mattamala, Nived Chebrolu, Cesar Cadena, Maurice Fallon, and Marco Hutter. 2023. Fast Traversability Estimation for Wild Visual Navigation. arXiv:2305.08510 [cs.RO] `https://arxiv.org/abs/2305.08510`

Jason Gibson, Anoushka Alavilli, Erica Tevere, Evangelos A. Theodorou, and Patrick Spieler. 2024. Dynamics Modeling using Visual Terrain Features for High-Speed Autonomous Off-Road Driving. arXiv:2412.00581 [cs.RO] `https://arxiv.org/abs/2412.00581`

Tyler Han, Alex Liu, Anqi Li, Alex Spitzer, Guanya Shi, and Byron Boots. 2024. Model Predictive Control for Aggressive Driving Over Uneven Terrain. arXiv:2311.12284 [cs.RO] `https://arxiv.org/abs/2311.12284`

James Harrison, Apoorva Sharma, and Marco Pavone. 2018. Meta-Learning Priors for Efficient Online Bayesian Regression. arXiv:1807.08912 [cs.RO] `https://arxiv.org/abs/1807.08912`

Gabriel M. Hoffmann, Claire J. Tomlin, Michael Montemerlo, and Sebastian Thrun. 2007. Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing. In *2007 American Control Conference*. 2296–2301. `https://doi.org/10.1109/ACC.2007.4282788`

Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. 2022. Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo. arXiv:2212.00541 [cs.RO] `https://arxiv.org/abs/2212.00541`

Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. 2019. Learning-Based Model Predictive Control for Autonomous Racing. *IEEE Robotics and Automation Letters* 4, 4 (2019), 3363–3370. `https://doi.org/10.1109/LRA.2019.2926677`

Donald E. Kirk. 2004. *Optimal Control Theory: An Introduction* (reprint of 1970 ed. ed.). Dover Publications, Mineola, NY.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. 2021. RMA: Rapid Motor Adaptation for Legged Robots. arXiv:2107.04034 [cs.LG] `https://arxiv.org/abs/2107.04034`

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. 2018. Model-Ensemble Trust-Region Policy Optimization. arXiv:1802.10592 [cs.LG] `https://arxiv.org/abs/1802.10592`

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. arXiv:1612.01474 [stat.ML] `https://arxiv.org/abs/1612.01474`

Dženan Lapandić, Fengze Xie, Christos K. Verginis, Soon-Jo Chung, Dimos V. Dimarogonas, and Bo Wahlberg. 2024. Meta-Learning Augmented MPC for Disturbance-Aware Motion Planning and Control of Quadrotors. arXiv:2410.06325 [cs.RO] `https://arxiv.org/abs/2410.06325`

Jacob Levy, Jason Gibson, Bogdan Vlahov, Erica Tevere, Evangelos Theodorou, David Fridovich-Keil, and Patrick Spieler. 2025. Meta-Learning Online Dynamics Model Adaptation in Off-Road Autonomous Driving. arXiv:2504.16923 [cs.RO] `https://arxiv.org/abs/2504.16923`

Elena Sorina Lupu, Fengze Xie, James A. Preiss, Jedidiah Alindogan, Matthew Anderson, and Soon-Jo Chung. 2024. MAGIC-VFM: Meta-learning Adaptation for Ground Interaction Control with Visual Foundation Models. arXiv:2407.12304 [cs.RO] `https://arxiv.org/abs/2407.12304`

Gabriel B. Margolis, Xiang Fu, Yandong Ji, and Pulkit Agrawal. 2023. Learning to See Physical Properties with Active Sensing Motor Policies. arXiv:2311.01405 [cs.RO] `https://arxiv.org/abs/2311.01405`

Matías Mattamala, Jonas Frey, Piotr Libera, Nived Chebrolu, Georg Martius, Cesar Cadena, Marco Hutter, and Maurice Fallon. 2024. Wild Visual Navigation: Fast Traversability Learning via Pre-Trained Models and Online Self-Supervision. arXiv:2404.07110 [cs.RO] `https://arxiv.org/abs/2404.07110`

Anusha Nagabandi, Guangzhao Yang, Thomas Asmar, Ravi Pandya, Gregory Kahn, Sergey Levine, and Ronald S. Fearing. 2018. Learning Image-Conditioned Dynamics Models for Control of Under-actuated Legged Millirobots. arXiv:1711.05253 [cs.RO] `https://arxiv.org/abs/1711.05253`

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. 2024. DINOv2: Learning Robust Visual Features without Supervision. arXiv:2304.07193 [cs.CV] `https://arxiv.org/abs/2304.07193`

Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. 2022. Neural-Fly enables rapid learning for agile flight in strong winds. *Science Robotics* 7, 66 (May 2022). `https://doi.org/10.1126/scirobotics.abm6597`

Spencer M. Richards, Navid Azizan, Jean-Jacques Slotine, and Marco Pavone. 2021. Adaptive-Control-Oriented Meta-Learning for Nonlinear Systems. arXiv:2103.04490 [cs.RO] `https://arxiv.org/abs/2103.04490`

Spencer M. Richards, Navid Azizan, Jean-Jacques Slotine, and Marco Pavone. 2022. Control-oriented meta-learning. arXiv:2204.06716 [cs.RO] `https://arxiv.org/abs/2204.06716`

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. 2024. gsplat: An Open-Source Library for Gaussian Splatting. arXiv:2409.06765 [cs.CV] `https://arxiv.org/abs/2409.06765`

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. 2020. MOPO: Model-based Offline Policy Optimization. arXiv:2005.13239 [cs.LG] `https://arxiv.org/abs/2005.13239`

A Implementation Details

A.1 Deriving Linear Feedback Controller

We derive the linear feedback controller in Eq. 10 as the closed-form solution to the quadratic tracking problem using the linearized error dynamics $e_{k+1} = A_k e_k + B_k \Delta U_k$ where $\Delta U_k = U_k - \overline{U}_k$:

$$\begin{aligned} &\min_{\Delta U_{0:N-1}} \sum_{k=0}^{N} ||e_k||_Q^2 + \sum_{k=0}^{N-1} ||\Delta U_k||_R^2 \\ &\text{s.t. } e_{k+1} = A_k e_k + B_k \Delta U_k, \quad \forall k = 0 : N-1 \end{aligned} \tag{16}$$

which reduces to linear quadratic regulation (LQR) Kirk (2004) and can be solved in closed-form, admitting a general solution in the form of a linear time-varying feedback policy $\Delta U_k = K_k e_k$ i.e., Eq. 10. The feedback gains $\{K_k\}_{k=0}^{N-1}$ are obtained via the backwards Riccati recursion:

$$P_N = Q_N \tag{17}$$

$$K_k = -\left(R + B_k^T P_{k+1} B_k\right)^{-1} \left(B_k^T P_{k+1} A_k\right) \tag{18}$$

$$P_k = Q_k + A_k^T P_{k+1} A_k - \left(A_k^T P_{k+1} B_k + S_k\right) \left(R + B_k^T P_{k+1} B_k\right)^{-1} \left(B_k^T P_{k+1} A_k\right) \tag{19}$$

We use $Q = \text{diag}(1, 1, 0, 0, 0, 0)$ as in the original policy and $R = \text{diag}(0.0001, 0.0001)$.

A.2 Dynamics Model Architecture

For the terrain latent model h we use an MLP of shape (384,50,25,3) with GeLU activations. For the force prediction model g we use an MLP of shape (8, 25, 15) with GeLU activations. We use $\gamma = 250$ for interpolation. We pass in to DINO images of size (90, 160). We train the models for 50 epochs with batch size 30 using Adam with default pytorch arguments.

A.3 MPC Solver Details

For the sampling-based solver developed in Howell et al. (2022), we generate at each time 1000 action spline perturbations using 3 knot points with perturbation variance $\Sigma = \text{diag}(0.1, 0.1)$. Of the 1000, we randomly set 1% to reset to just the perturbation, instead of adding to the previously applied action sequence.