MAGIC-VFM - Meta Adaptive Control for Ground Vehicles with Visual Foundation Models

Elena Sorina Lupu*, Fengze Xie*, James A. Preiss, Matthew Anderson, Jedidiah Alindogan, Soon-Jo Chung

***Abstract*—Planning and control of ground vehicles are challenging because of complex dynamic interactions with the terrain. Therefore, accurate modeling of terrain interaction forces is important to optimize their driving performance. We present an offline meta-learning algorithm to construct a rapidly-tunable model of residual dynamics and disturbance using both visual foundation models and vehicle states. This model is then integrated with composite adaptive control for the control matrix to adapt to changes in both the terrain and vehicle dynamics conditions in real time. We provide mathematical guarantees of stability and robustness for this controller that provides rapid adaptation for the last layer of the deep neural network, which encodes the terrain disturbance. The effectiveness of our meta-adaptive algorithm is demonstrated through results of simulation and experimentation using a tracked vehicle. In particular, the experimental results show its superior performance outdoors on different slopes with varying slippage and track degradation disturbances. We also compare our controller against an adaptive controller that does not use the visual foundation model terrain features, thereby showing significant improvement over the baseline in both hardware experimentation and simulation.**

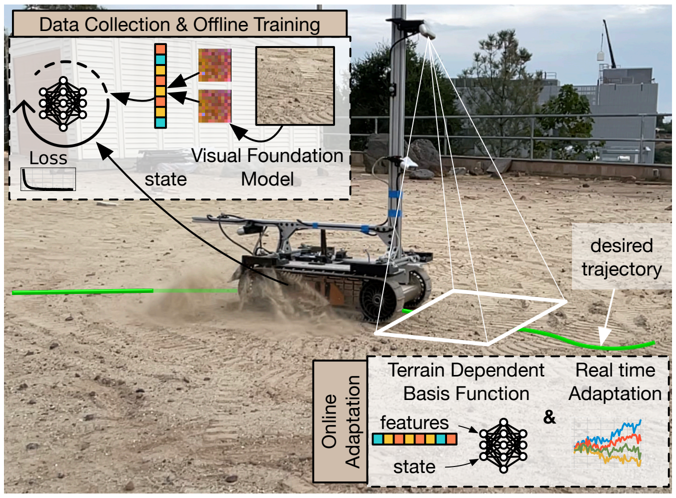


Fig. 1: **Concept overview.** The basis function for MAGIC$^{\mathrm{VFM}}$ adaptive control is a deep neural network learned using a meta-learning method that takes as input both robot's state and visual information processed through a visual foundation model. The adaptive control then updates this basis function of the control matrix in real time for superior terrain and disturbance adaptation performance.

Supplementary Material

Video demonstration of the proposed meta-adaptive control for ground vehicles with visual foundation models project: https://youtu.be/5mpEVbIzybM.

I. Introduction

Autonomous Ground Vehicles (AGVs) are gaining popularity across numerous domains including agriculture [1]–[3], wilderness search and rescue missions [4]–[7], and planetary exploration [8]. In many of these scenarios, the AGVs operate on rugged surfaces where the ability to follow a desired trajectory will be degraded. To reliably operate in these environments with minimal human intervention, AGVs must understand the environment and adapt to it in real time. Slippage is one of the primary challenges ground vehicles encounter while operating on loose terrains. For rovers exploring other planets, slippage can slow down their progress and even halt their scientific objectives. For instance, the Opportunity rover recorded significant slippage and sinking of its wheels during the Mars day 2220 [9] while traversing sand ripples. During its climb, the slip, calculated based on visual odometry [10], was high, and thus the drive was halted and replanned by the ground operators.

To better understand the effects of terradynamics, researchers have designed sophisticated models [11] that inform the design, simulation, and control of ground vehicles. However, these models have numerous assumptions and are often limited when the vehicles are operated at their performance boundaries (e.g., steering at high speeds, traversing highly uneven terrain, and instances of non-uniform resistive forces like stumps and stones). In addition, designing controllers that consider these complex models is challenging. For control, kinematic models, such as Dubins or other simplified models, are often employed due to their simplicity and intuitive understanding. However, these models are not able to capture the complicated dynamics between the vehicle and the ground, nor other disturbances such as internal motor dynamics or wheel or track degradation. To increase the performance of ground vehicles, more comprehensive models are necessary.

Vehicle controllers that can stabilize a ground vehicle and track desired trajectories amidst a variety of disturbances are crucial for achieving optimal vehicle performance. Oftentimes, the bottleneck is not in the controller design per se, but rather in the choice and complexity of the model utilized within the controller. In recent years, reinforcement learning (RL) has shown significant promise in facilitating the development of efficient controllers through experiential learning [12]–[14]. The combination of meta-learning [15]–[22] and adaptive control [23]–[31] demonstrates considerable potential in accurately estimating unmodeled dynamics, effectively addressing

*Co-first authors. This work was supported by Defense Advanced Research Projects Agency (DARPA) Learning Introspective Control (LINC) and Amazon AI for Science.

domain shift challenges and real-time adaptation to new environments [18], [32]–[34]. Despite this progress, incorporating a suitable model into a controller/policy is still an active area of research, especially when combined with theoretical and safety guarantees.

Learning sophisticated unmodeled dynamics based only on a limited set of vehicle states is ill-posed given that the operating environment is infinite dimensional. To accurately represent a complete dynamics model, including learned residual terms for control, it is imperative to leverage as much information about the environment as possible. For instance, visual information can inform the model about the type of terrain in which the vehicle is operating. Previous work includes segmentation-based models that assign a discrete terrain type to each area in the image. This information is further employed in planning and control [35]. However, in off-road applications, categorizing terrains into a limited number of classes such as snow, mud, sand, etc. is not sufficient. There are infinite subcategories within each terrain type, each presenting distinct effects on the vehicle. In addition, two terrains can appear similar, but induce different dynamic behaviors on the robot (e.g., deep and shallow sand). Therefore, finding the right robust representation of the environment is indispensable for vehicle control over complex terrain.

A. *Contributions*

To address these limitations, we introduce MAGIC$^{\text{VFM}}$ (**M**eta-learning **A**daptation for **G**round **I**nteraction **C**ontrol with **V**isual **F**oundation **M**odels), an approach that integrates a Visual Foundation Model (VFM) with meta-learning and adaptive control, thereby enabling ground vehicles to navigate and adapt to complex terrains in real time. Our method is suited for any ground vehicle equipped with the following: 1) sensors that measure the robot's internal state, 2) exteroceptive sensors that can capture the terrain such as cameras, 3) the availability of a pre-trained VFM, and 4) the necessary computation hardware to evaluate the VFM in real-time. Our contributions are:

- the first stable *learning-based adaptive controller* that incorporates *visual foundation models* for terrain adaptation;
- an offline meta-learning algorithm that uses continuous trajectory data to train and learn the terrain disturbance as a function of visual terrain information and vehicle states;
- mathematical guarantees of exponential stability for adaptive control with visual and state input that works in conjunction with our deep meta-learning offline training algorithm;
- the development of a position, attitude, and velocity tracking control formulation with the control influence matrix adaptation that can handle a variety of other perturbations in real-time such as unknown time-varying track or motor degradation, arbitrary time-varying disturbances, and model uncertainties.

We validate the effectiveness of our method both through simulation and in hardware, demonstrating its performance outdoors, on slopes with different slippage, as well as under track degradation disturbances.

B. *Paper Organization*

The paper is organized as follows: Section II provides a review of existing literature. In Section III, both our offline meta-learning algorithm and the online adaptation algorithm are presented. In Section IV, we present the tracked vehicle model, as well as the adaptive controller with its theoretical convergence and robustness guarantees. In Section V, we analyze the VFM output for terrain, particularly in relation to our learning-based adaptive controller. In Section VI, we validate the algorithm using simulation results and continue with experimental validation in Section VII. The concluding remarks are given in Section VIII.

C. *Notation*

Unless otherwise noted, all vector norms are Euclidean and all matrix norms are the Euclidean operator norm. We denote the floor operator by $\lfloor\cdot\rfloor$. Given $\mathbf{A} \in \mathbb{R}^{n\times m\times p}$ and $\mathbf{b} \in \mathbb{R}^p$, the notation $(\mathbf{A}\mathbf{b})$ is defined as $\sum_{i=1}^{p} \mathbf{A}_i b_i$. The notation $\|\mathbf{x}\|_{\mathbf{P}}$ for positive semi-definite matrix $\mathbf{P}$ defines the weighted inner product $\sqrt{\mathbf{x}^\top \mathbf{P}\mathbf{x}}$. For a function $f : X \mapsto Y$ where X and Y are metric spaces with metrics d_X and d_Y, we define $\|f\|_{\text{Lip}} = \max_{x,x'\in X} d_Y(f(x), f(x'))/d_X(x, x')$. For a measurable set X, we denote the set of probability measures on X by $\triangle X$, and if a uniform distribution on X exists, we denote it by $\mathcal{U}X$. When clear from context, we overload the notation $[i, j]$ to denote the integer sequence $i, \ldots, j$.

II. Related Work

The term *meta-learning*, first coined in [36], most often refers to learning protocols in which there is an underlying set of related learning tasks/environments, and the learner leverages data/computation from previously seen tasks to adapt rapidly to a new task [37]–[40]. The goal is to adapt more rapidly than would be possible for a standard learning algorithm presented with the new task in isolation. In robotics, meta-learning has been used to accurately adapt to highly dynamic environments [18], [32]–[34]. Online meta-learning [27], [41]–[43] includes two phases: offline meta-training and online adaptation. In the offline phase, the goal is to learn a model that performs well across all environments using a meta-objective. Given limited real-world data, the online adaptation phase aims to use online learning, such as adaptive control [23], to adapt the offline-learned model to a new environment in real time.

Some examples of meta-learning algorithms from literature are Model-Agnostic Meta-Learning (MAML) [44] with its online extension [43], Meta-learning via Online Changepoint Analysis (MOCA) [45], and Domain Adversarially Invariant Meta-Learning (DAIML) used in Neural-Fly [27]. For task-centered datasets, MAML [44] trains the parameters of a model to achieve optimal performance on a new task with minimal data, by updating these parameters through one or more gradient steps based on that task's dataset. In continuous problems, tasks often lack clear segmentation, resulting in the agent being unaware of task transitions. Therefore, MOCA [45] proposes a task unsegmented meta-learning via online change-point analysis. Designed mainly for aerial vehicles, although applicable to other problems, as well, DAIML [27] proposes an online meta-learning-based approach that allows rapid online adaptation. Here, a shared representation in different wind conditions is learned offline, with a linear, low-dimensional, wind-specific part updated online through adaptive control.

Our method builds on the previous works on the integration of adaptive control and offline meta-learning to build a comprehensive model for ground vehicles. We develop a meta-learning algorithm that uses continuous trajectories from a robot driving on different terrains to learn a representation of the dynamics residual common across these terrains. This representation is a deep neural network (DNN) that encodes the terrain information through vision, together with a set of linear parameters that adapt online, at runtime. These linear parameters can be interpreted as the last layer of the DNN [27] and will be terrain independent, but encapsulate the remaining disturbances not captured during training, such as track or wheel degradation or unmodeled internal dynamics.

A. Embedding Visual Information in Classical Control and Reinforcement Learning

One of the early works on including visual information for control is visual servoing [46], a technique mainly used for robot manipulation. Recently, vision-based reinforcement learning (VRL) has demonstrated the capability to control agents in simulated environments [47], as well as robots in real environments [48]–[50]. This capability is achieved by leveraging high-fidelity models in robotics simulators [51], [52] and techniques to bridge the sim-to-real gap of the learned policy [14]. Nevertheless, a notable limitation of VRL is that the generated policy remains uninterpretable and does not have safety and robustness guarantees. To address the uninterpretability aspect, recent advancements in Inverse Reinforcement Learning (IRL) offer promising methods for interpreting terrain traversability as a reward map, thus enhancing the understanding of the environments [53], [54]. Despite progress in combining vision with RL, incorporating a suitable terrain model into a policy is still an open area of research, especially when combined with theoretical guarantees and safety properties.

To this end, we derive a nonlinear adaptive tracking controller for AGVs that uses a learned ground model with vision information, represented in the control influence matrix. Our method processes camera images that are then passed through a VFM to synthesize the relevant features. These features, together with the robot's state, are incorporated into the ground model learned offline using meta-learning. For this adaptive controller, we prove exponential convergence to a bounded error ball.

B. Visual Foundation Models in Robotics

A foundation model is a large-scale machine learning model trained on a broad dataset that can be adapted, fine-tuned, or built upon for a variety of applications. Self-supervised learned VFMs, such as Dino and DinoV2 [55], are foundation models that are based on visual transformers [56], [57]. These models are trained to perform well on several downstream tasks, including image classification, semantic segmentation, and depth estimation. In robotics, these foundation models are starting to gain popularity in tasks such as image semantic segmentation [58], [59] and traversability estimation [54]. One of their key advantages lies in the robustness against variations in lighting and occlusions [60], as well as their ability to generalize well across different images of the same context. By consuming raw images as inputs, these self-supervised learning foundation models possess the potential to learn all-purpose visual features if pre-trained on a large quantity of data.

C. Adaptation to Ground Disturbances

Adaptive control [61] is a control method with provable convergence guarantees in which a set of linear parameters are adapted online to compensate for disturbances at runtime. Typically, these parameters are multiplied by a basis function, which can be constant (as in the case of integral control), derived from physics [62], or represented using Radial Basis Functions (RBFs) [63] or Deep Neural Networks [27].

Ground vehicles (including cars, tracked vehicles, and legged robots) should be adaptive to changes in the terrain conditions. This adaptability is essential for optimal performance and safety in diverse environments [64]–[69] in which the robot operates. In [70], an adaptive energy-aware prediction and planning framework for vehicles navigating over terrains with varying and unknown properties has been proposed and demonstrated in simulation. [71] proposes a deep meta-learning framework for learning a global terrain traversability prediction network that is integrated with a sampling-based model predictive controller, while [72] develops a probabilistic traction model with uncertainty quantification using both semantic and geometric terrain features.

We establish an adaptive controller that can deal with a broad range of real-time perturbations, such as unknown time-varying track or motor degradation, under controllability assumptions, arbitrary time-varying disturbances, and model uncertainties. From an adaptive control derivation standpoint, this work can be viewed as a generalization or improvement of [27], [73] with a new control matrix adaptation method using visual information and improved stability results. This adaptive capability acts as an enhancement to the offline trained basis function.

III. Methods

A. Residual Dynamics Representation using VFM

We consider an uncertain dynamical system model:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) + \mathbf{d}, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ denotes the state, $\mathbf{u} \in \mathbb{R}^m$ denotes the control input, $\mathbf{f} : \mathbb{R}^{n+m+1} \mapsto \mathbb{R}^n$ denotes the nominal dynamics

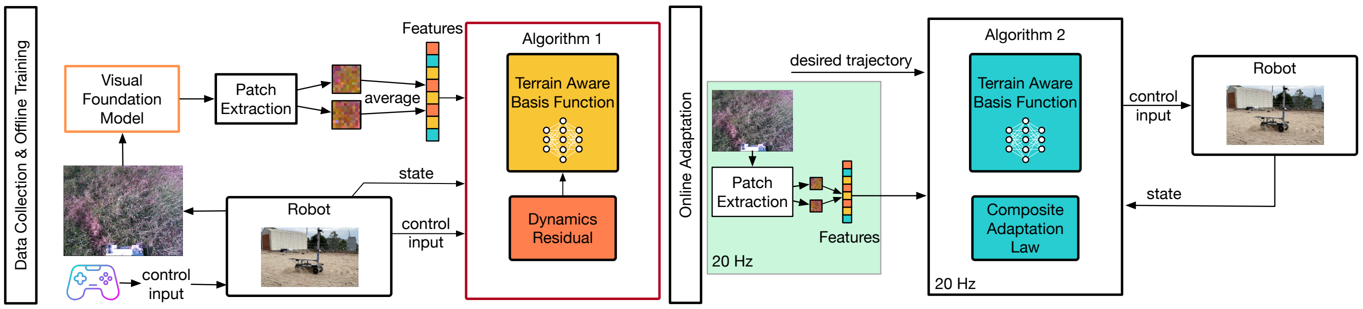


Fig. 2: Terrain-aware Architecture: offline data collection and training (Algorithm 1), followed by real-time adaptive control (Algorithm 2) running onboard the robot.

model, and $\mathbf{d}$ is an unknown disturbance that is possibly time-varying and state- and environment-dependent. Our algorithm approximates $\mathbf{d}$ by:

$$\mathbf{d} = \mathbf{\Phi}(\mathbf{x}, \mathbf{u}, \boldsymbol{\mathcal{E}})\boldsymbol{\theta}(t) + \boldsymbol{\delta}, \tag{2}$$

where $\boldsymbol{\delta}$ is a representation error, $\boldsymbol{\theta} = [\theta_1 \dots \theta_{n_\theta}]^\top \in \mathbb{R}^{n_\theta}$ denotes a vector of linear parameters that are adapted online by our algorithm, and $\boldsymbol{\mathcal{E}} \in \mathbb{R}^p$ is a feature vector representation of the terrain surrounding the robot computed by a VFM. From the perspective of our algorithm, the precise method by which $\boldsymbol{\mathcal{E}}$ is computed is not important. We can think of $\boldsymbol{\mathcal{E}}$ as computed by some arbitrary function from the robot's sensors to $\mathbb{R}^p$. We give details on the particular form of $\boldsymbol{\mathcal{E}}$ used in our empirical sections (Sections V-VII). The feature mapping $\mathbf{\Phi}(\mathbf{x}, \mathbf{u}, \boldsymbol{\mathcal{E}}) : \mathbb{R}^{n+m+p} \mapsto \mathbb{R}^{n \times n_\theta}$ is learned in the offline training phase of our algorithm. Our method supports arbitrary parameterized families of continuous functions, but in practice we focus on the case where $\mathbf{\Phi}$ is a DNN. In this case, $\boldsymbol{\theta}$ can be regarded as the weights of the last layer of the DNN (2), which continuously adapts in real-time. The online adaptation is necessary in real scenarios, because two terrains might have the same terrain representation $\boldsymbol{\mathcal{E}}$, but induce different dynamic behaviors onto the robot, as well as for other types of disturbances not captured in $\mathbf{\Phi}$.

Further, we assume that the disturbance $\mathbf{d}$ is affine in the control input $\mathbf{u}$:

$$\mathbf{d} \approx \mathbf{\Phi}^{\mathbf{w}}(\mathbf{x}, \boldsymbol{\mathcal{E}})\boldsymbol{\theta}\mathbf{u} = \sum_{i=1}^{n_\theta} \theta_i \mathbf{\Phi}_i^{\mathbf{w}}(\mathbf{x}, \boldsymbol{\mathcal{E}})\mathbf{u} = \mathbf{H}(\mathbf{\Phi}^{\mathbf{w}}, \mathbf{u})\boldsymbol{\theta}, \tag{3}$$

in which the dependence on a parameter vector $\mathbf{w}$, for example the DNN weights, is made explicit, and the basis function has the form $\mathbf{\Phi}^{\mathbf{w}}(\mathbf{x}, \boldsymbol{\mathcal{E}}) : \mathbb{R}^{n+p} \mapsto \mathbb{R}^{n \times m \times n_\theta}$ with the individual matrix-valued components denoted $\mathbf{\Phi}_i^{\mathbf{w}}(\mathbf{x}, \boldsymbol{\mathcal{E}}) : \mathbb{R}^{n+p} \mapsto \mathbb{R}^{n \times m}$. We define $\mathbf{H} \in \mathbb{R}^{n \times n_\theta}$, where $\mathbf{H} = [\mathbf{h}_1 \dots \mathbf{h}_{n_\theta}]$, with the columns $\mathbf{h}_i = \mathbf{\Phi}_i^{\mathbf{w}}\mathbf{u}$, for each $i \in [1, n_\theta]$. The control affine assumption is motivated by two reasons. First, in our main application of ground vehicles with desired-velocity inputs, input-affine disturbances more accurately capture terrain interactions such as slippage (Section IV-C1), internal dynamics, and wheel or track degradations. Second, it simplifies the exponential convergence proof for our adaptive controller given in Theorem 1. However, we emphasize that Theorem 1 can be extended to more general forms of

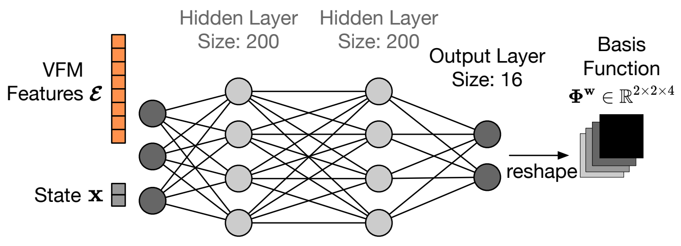


Fig. 3: The structure of the DNN used for the basis function $\mathbf{\Phi}^{\mathbf{w}}$ in the controller synthesis from (21), (23) applied to a tracked vehicle.

disturbances using techniques from contraction theory [74], [75]. Lastly, we define the dynamics residual derivative $\mathbf{y}$ as $\dot{\mathbf{x}} - \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$.

B. Offline Meta-Learning Phase

In Fig. 2, we give an overview of the structure of our proposed solution to learn offline the basis function (3) and to make real-time adjustments using composite adaptive control (Section III-C). Our method is divided into two steps. First, the robot captures relevant ground information during offline data collection, followed by training a DNN with terrain and state information to approximate the residual dynamics $\mathbf{y}$. Second, the trained model is deployed onboard the robot and updated online to compensate for the residual dynamics not captured in offline training.

1) Dataset: To learn the basis function in (3), we collect a dataset of robot operation on a diverse set of terrains. The dataset includes paired ground images from the onboard camera and state information measured using onboard sensors. The images are processed through a VFM, resulting in the $\boldsymbol{\mathcal{E}}$ representation as discussed in Section III-A.

This dataset contains $N \in \mathbb{N}$ trajectories. Each trajectory is an uninterrupted driving session on the order of a few minutes. Therefore, a single trajectory may contain significant dynamics-altering terrain transitions, such as between grass and concrete, but it will not contain dramatic transitions such as from a desert to a forest, or from midday to night. For notational simplicity only, we assume all trajectories have equal length $\ell \in \mathbb{N}$. Let $\mathbf{x}_t^n, \mathbf{u}_t^n, \boldsymbol{\mathcal{E}}_t^n, \mathbf{y}_t^n$ respectively denote the t^{th} state, input, VFM feature, and residual dynamics derivative of the n^{th} trajectory.

Algorithm 1 Offline Meta-Learning with Continuous Trajectories

```
1: Input
2:     Dataset of $N$ trajectories of length $\ell$.
3:     Window length distribution $L$.
4:     Regularization target $\boldsymbol{\theta}_\mathrm{r}$ and weight $\lambda_\mathrm{r}$.
5:     Minibatch size $K$.
6:     Initial DNN weights $\mathbf{w}$.
7: Output: Final optimized DNN weights $\mathbf{w}$.
8: while not converged do
9:     Sample (with replacement) size-$K$ minibatches of:
10:        - trajectory indices $\{n_k\}_{k=1}^K$,
11:        - window lengths $\{\hat{\ell}_k\}_{k=1}^K$,
12:        - start times $\{s_k\}_{k=1}^K$.
13:    Solve (4) for each $\theta^\star_{n_k,\hat{\ell}_k,s_k}$ in the minibatch
14:        (in closed form, allowing $J_{n_k,\hat{\ell}_k,s_k}$ gradient flow[1]).
15:    Take optimizer step on $\mathbf{w}$ w.r.t minibatch cost (5):
               $\sum_{k=1}^K J_{n_k,\hat{\ell}_k,s_k}(\mathbf{w})$.
16:    Spectral Normalizaton: $\mathbf{W}_i \leftarrow \frac{\mathbf{W}_i}{\|\mathbf{W}_i\|}$ for all $i \in [d]$.
17: end while
```

2) Model Architecture: In designing the parameterized function class for $\boldsymbol{\Phi}^\mathbf{w}$, we prioritize simplicity and efficiency to enable inference at high frequency for effective real-time control. Therefore, we select a fully connected DNN with two hidden layers (Fig. 3). We employ layer-wise spectral normalization to constrain the Lipschitz constant of the DNN. Spectral normalization is crucial for ensuring smooth control outputs and limits pathological behavior outside the training domain [74]. Additionally, our network combines both the robot's state and visual features from the VFM. Details of spectral normalization are given in Section III-B3.

3) Optimization: Our method is built around the following assumption: two terrains with similar visual features $\boldsymbol{\mathcal{E}}$ will usually, but not always, induce similar dynamics. We account for this with a meta-learning method that allows the linear part $\boldsymbol{\theta}$ to vary over the training data while the feature mapping weights $\mathbf{w}$ remain fixed. In particular, we assume that the linear part $\boldsymbol{\theta}$ is slowly time-varying within a single trajectory in the training data, but may change arbitrarily much between two trajectories.

The slowly time-varying assumption implies that within a sufficiently short window into a full trajectory, $\boldsymbol{\theta}$ is approximately constant. Therefore, we optimize $\mathbf{w}$ for data-fitting accuracy when the best-fit constant $\boldsymbol{\theta}$ is computed for random short windows into the trajectories. Due to our linear adaptation model structure, we observe that for a particular trajectory index $n \in [1 \ldots N]$, window length $\hat{\ell} \in [1 \ldots \ell]$, and starting timestep $s \in [1 \ldots \ell - \hat{\ell} + 1]$, the best-fit value:

$$\boldsymbol{\theta}^\star_{n,\hat{\ell},s}(\mathbf{w}) := \operatorname*{argmin}_{\boldsymbol{\theta}} \sum_{t=s}^{s+\hat{\ell}-1} \|\mathbf{y}_t^n - (\boldsymbol{\Phi}^\mathbf{w}(\mathbf{x}_t^n, \boldsymbol{\mathcal{E}}_t^n)\boldsymbol{\theta})\mathbf{u}_t^n\|_2^2 + \lambda_r \|\boldsymbol{\theta} - \boldsymbol{\theta}_r\|_2^2, \tag{4}$$

is the solution to an L_2-regularized linear least-squares problem and is a closed-form, continuous function of the feature mapping parameter $\mathbf{w}$. We can now define our overall optimization objective. Let L denote a distribution over trajectory window lengths: $L \in \triangle[1, \ell]$. We minimize the meta-objective:

$$\begin{aligned} J(\mathbf{w}) &= \mathop{\mathbb{E}}_{n,\hat{\ell},s} \left[\sum_{t=s}^{s+\hat{\ell}-1} \left\| \mathbf{y}_t^n - \left(\boldsymbol{\Phi}^\mathbf{w}(\mathbf{x}_t^n, \boldsymbol{\mathcal{E}}_t^n)\boldsymbol{\theta}^\star_{n,\hat{\ell},s}(\mathbf{w}) \right) \mathbf{u}_t^n \right\|_2^2 \right] \\ &:= \mathop{\mathbb{E}}_{n,\hat{\ell},s} \left[J_{n,\hat{\ell},s}(\mathbf{w}) \right], \end{aligned} \tag{5}$$

where the expectation is shorthand for $n \sim \mathcal{U}[1, N]$, $\hat{\ell} \sim L$, and $s \sim \mathcal{U}[1, \ell - \hat{\ell} + 1]$. By incorporating the closed-form computation of $\boldsymbol{\theta}^\star_{n,\hat{\ell},s}$ in the computational graph of our optimization, as opposed to treating the trajectories of $\boldsymbol{\theta}$ as optimization variables, we obtain a simpler algorithm.

Our offline training procedure is given in Algorithm 1. It consists of stochastic first-order optimization on the objective $J(\mathbf{w})$ and spectral normalization to enforce the Lipschitz constraint on $\boldsymbol{\Phi}$. In particular, let $\mathbf{w} = (\mathbf{W}_1, \ldots, \mathbf{W}_d, \overline{\mathbf{w}})$, where $\mathbf{W}_1, \ldots, \mathbf{W}_d$ are the dimensionally compatible weight matrices of the DNN, such that the product $\mathbf{W}_1 \cdots \mathbf{W}_d$ exists, and $\overline{\mathbf{w}}$ are the remaining bias parameters. It holds that $\|\boldsymbol{\Phi}^\mathbf{w}\|_\mathrm{Lip} \leq \|\mathbf{W}_1 \cdots \mathbf{W}_d\|$ for neural networks with 1-Lipschitz nonlinearities. Therefore, we can enforce that $\|\boldsymbol{\Phi}^\mathbf{w}\|_\mathrm{Lip} \leq 1$, by enforcing that $\|\mathbf{W}_i\| \leq 1$ for all $i \in [1 \ldots n]$. This is implemented in Line 16 of Algorithm 1. Note that finding less conservative ways to enforce $\|\boldsymbol{\Phi}^\mathbf{w}\|_\mathrm{Lip} \leq 1$ for neural networks is an active area of research.

For the remaining sections, the parameters $\mathbf{w}$ of $\boldsymbol{\Phi}$ are fixed. Therefore, we drop the superscript and refer to $\boldsymbol{\Phi}^\mathbf{w}$ as $\boldsymbol{\Phi}$, for simplicity of notation.

C. Online Adaptation and Control

This section introduces the online adaptation running onboard the robot to adapt the linear part $\boldsymbol{\theta}$ of the dynamics model. The algorithm is given in Algorithm 2. The parameter $\boldsymbol{\theta}$ is initialized with the regularization target $\boldsymbol{\theta}_\mathrm{r}$. Then, in each cycle of the main loop, the robot processes the data from its visual sensor through the VFM to generate the feature vector $\boldsymbol{\mathcal{E}}$. The feature mapping $\boldsymbol{\Phi}$ is then evaluated using the robot's current state and $\boldsymbol{\mathcal{E}}$. This information is passed to the adaptive controller, which ensures the robot tracks its trajectory and performs online adaptation. In this way, model mismatches and other disturbances not captured during training can be adapted in real-time.

For each sampled measurement (interaction with the environment), the adaptation parameter vector is updated using the composite adaptation rule in (6). Here we give an intuition for the role of each term; we show exponential stability in Section IV-C. The first term in (6) implements the so-called "exponential forgetting" to allow $\boldsymbol{\theta}$ to change more rapidly when the best-fit parameters are time-varying. The second term is gradient descent on the $\mathbf{R}^{-1}$-weighted squared prediction error $\frac{1}{2}\|\mathbf{y} - (\boldsymbol{\Phi}\hat{\boldsymbol{\theta}})\mathbf{u}\|^2_{\mathbf{R}^{-1}}$ with respect to $\boldsymbol{\theta}$. The adaptive gain can be defined from least-squares with exponential forgetting [62],

[1]We use the machine learning framework PyTorch that implements the linear least squares solution with gradient flow.

Algorithm 2 Rapid Online Adaptation for Model Mismatch and Tracking Error

```
 1: Input
 2:     Optimized feature mapping $\mathbf{\Phi}$ from Algorithm 1.
 3:     Importance weight for prediction $\mathbf{R}$.
 4:     Damping constants $\lambda$ and $q$.
 5:     Initial adaptive gain $\mathbf{\Gamma}_0$.
 6:     Reference trajectory $\mathbf{x}_r$.
 7: Initialize $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{\mathrm{r}}$.
 8: while running do
 9:     Evaluate the VFM and get the feature vector $\boldsymbol{\mathcal{E}}$.
10:     Get state $\mathbf{x}$ and evaluate $\mathbf{\Phi}(\mathbf{x}, \boldsymbol{\mathcal{E}})$.
11:     Compute the error vector $\mathbf{s} = \mathbf{x} - \mathbf{x}_r$.
12:     Compute the control input $\mathbf{u}$ using (21).
13:     Compute adaptation parameter vector derivative $\dot{\hat{\boldsymbol{\theta}}}$:
            $\dot{\hat{\boldsymbol{\theta}}} = -\lambda\hat{\boldsymbol{\theta}} - \mathbf{\Gamma}\mathbf{H}^\top\mathbf{R}^{-1}(\mathbf{H}\hat{\boldsymbol{\theta}} - \mathbf{y}) + \mathbf{\Gamma}\mathbf{H}^\top\mathbf{s}.$   (6)
14:     Compute adaptation gain derivative $\dot{\mathbf{\Gamma}}$.
15:     Integrate with system timestep $\Delta_t$:
            $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} + \Delta_t\dot{\hat{\boldsymbol{\theta}}}, \quad \mathbf{\Gamma} \leftarrow \mathbf{\Gamma} + \Delta_t\dot{\mathbf{\Gamma}}.$
16: end while
```

[76] or reminiscent of a Kalman Filter like in [27]. Thus, this composite adaptation is used to ensure both small tracking errors and low model mismatch. The stability and robustness properties of Algorithm 2 are presented in Section IV-C.

IV. System Modelling and Control Synthesis

We apply the methods from Algorithm 1 and 2 to a tracked vehicle (Fig. 4) with velocity input. A tracked vehicle uses skid-steering to maneuver over the ground, with its tracks moving at different speeds depending on the sprocket's angular velocity. Due to the slip between the sprocket and the tracks and between the tracks and the ground, modeling the full dynamics becomes very complex. We therefore derive its 3 degrees of freedom (DOF) dynamics model (12) with its corresponding simplified model of the form (13). To this simplified model, we apply an adaptive controller with learned ground information of the form (21),(23). Although particularized for a tracked vehicle, the controller can be easily modified for any ground vehicle with complex terrain interaction, internal unknown dynamics, and actuation malfunctions.

A. Tracked Vehicle Dynamics Model

We start by defining a fixed reference frame $\mathcal{I}$ and a moving reference frame $\mathcal{B}$ attached to the body of the tracked vehicle, as seen in Fig. 4. We assume the robot's center of mass coincides with the origin of $\mathcal{B}$. Next, consider the 3DOF dynamics model with the generalized coordinates $\mathbf{q} := [p_x^{\mathcal{I}}, p_y^{\mathcal{I}}, \psi] \in \mathbb{R}^3$, where $p_x^{\mathcal{I}}$ and $p_y^{\mathcal{I}}$ are the inertial positions and ψ is the yaw angle from $\mathcal{I}$ to $\mathcal{B}$, as follows:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{B}(\mathbf{q})\boldsymbol{\tau}_u + \mathbf{F}_r(\mathbf{q},\dot{\mathbf{q}}), \tag{7}$$

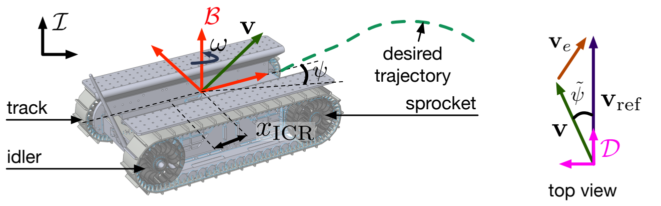


Fig. 4: On the left, the frames of reference for the tracked vehicle, its corresponding velocities, and the main driving components. On the right, a velocity vector diagram used for the proof of Theorem 2.

where $\mathbf{M} \in \mathbb{R}^{3\times3}$ is the inertia matrix, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{3\times3}$ is the Corriolis and centripetal matrix, $\mathbf{B}(\mathbf{q}) \in \mathbb{R}^{3\times2}$ is the control actuation matrix, $\mathbf{F}_r(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^3$ are the dissipative track forces due to surface-to-soil interaction, and $\boldsymbol{\tau}_u \in \mathbb{R}^2$ is the control torque.

Developing a tracking controller for the system modeled using (7) is difficult because the system is underactuated. To address this complexity, previous work [77], [78] introduced a nonholonomic constraint for (7), which reduces the number of state variables. Therefore, the following constraint constrains the ratio of the lateral body velocity ($\dot{p}_y^{\mathcal{B}}$) to the angular velocity (ω):

$$\dot{p}_y^{\mathcal{B}} + x_{\mathrm{ICR}}\omega = 0. \tag{8}$$

where x_{ICR} is the instantaneous center of rotation. We can embed this constraint into (7), as follows:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{B}(\mathbf{q})\boldsymbol{\tau}_u + \mathbf{F}_r(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{A}(\mathbf{q})^\top\lambda_c, \tag{9}$$

with λ_c is the Lagrange multiplier corresponding to the equality constraint in (8). By assuming x_{ICR} constant, $\mathbf{A}(\mathbf{q}) \in \mathbb{R}^{1\times3}$ is defined as follows, in which (8) in expressed in the $\mathcal{I}$ frame:

$$\begin{bmatrix} -\sin\psi & \cos\psi & x_{\mathrm{ICR}} \end{bmatrix} \begin{bmatrix} \dot{p}_x^{\mathcal{I}} \\ \dot{p}_y^{\mathcal{I}} \\ \omega \end{bmatrix} = \mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = 0 \tag{10}$$

To annihilate the constraint force from (9), an orthogonal projection operator $\mathbf{S}(\mathbf{q}) \in \mathbb{R}^{3\times2}$ is defined, whose columns are in the nullspace of $\mathbf{A}^\top(\mathbf{q})$, and thus $\mathbf{S}(\mathbf{q})^\top\mathbf{A}(\mathbf{q})^\top = \mathbf{0}$ [79], [80].

$$\mathbf{S}(\mathbf{q}) = \begin{bmatrix} \cos(\psi) & x_{\mathrm{ICR}}\sin(\psi) \\ \sin(\psi) & -x_{\mathrm{ICR}}\cos(\psi) \\ 0 & 1 \end{bmatrix} \tag{11}$$

We choose this projection operator conveniently to transform the velocities in the $\mathcal{I}$ frame to $\mathbf{v} = [v_x^{\mathcal{B}}, \omega]^\top$, with $v_x^{\mathcal{B}}$ being the projection of the inertial velocity onto the body x-forward axis. The reduced form can be written as [81]:

$$\begin{aligned} \dot{\mathbf{q}}(t) &= \mathbf{S}(\mathbf{q})\mathbf{v}(t), \\ \dot{\mathbf{v}}(t) &= \widetilde{\mathbf{M}}^{-1}(\widetilde{\mathbf{B}}(\mathbf{q})\boldsymbol{\tau}_u - \widetilde{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})\mathbf{v}(t) + \widetilde{\mathbf{F}}_r(\mathbf{q},\dot{\mathbf{q}})), \end{aligned} \tag{12}$$

with the reduced matrices:

$$\widetilde{\mathbf{M}} = \mathbf{S}^\top(\mathbf{q})\mathbf{M}\mathbf{S}(\mathbf{q}) = \begin{bmatrix} m & 0 \\ 0 & I_z + m x_{\mathrm{ICR}}^2 \end{bmatrix},$$
$$\widetilde{\mathbf{F}}_r = \mathbf{S}^\top(\mathbf{q})\mathbf{F}_r, \quad \widetilde{\mathbf{B}}(\mathbf{q}) = \mathbf{S}^\top(\mathbf{q})\mathbf{B}(\mathbf{q}),$$
$$\widetilde{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{S}^\top\mathbf{M}\dot{\mathbf{S}} = \begin{bmatrix} 0 & m x_{\mathrm{ICR}}\omega \\ -m x_{\mathrm{ICR}}\omega & 0 \end{bmatrix}.$$

where m is the mass of the robot and I_z is the inertia of the robot about the rotational degree of freedom. Note that $\dot{\widetilde{\mathbf{M}}} - 2\widetilde{\mathbf{C}}$ is skew-symmetric.

B. Simplified Vehicle Dynamics Model with Velocity Input

Due to the limited access to the robot's internal control software, specifically the absence of direct torque command capabilities, we are only able to utilize velocity input commands. Consequently, we have chosen to simplify the system in (12), as follows, where the velocity is modeled as a first-order time delay:

$$\begin{aligned}\dot{\mathbf{q}}(t) &= \mathbf{S}(\mathbf{q})\mathbf{v}(t),\\ \dot{\mathbf{v}}(t) &= \mathbf{A}_\mathrm{n}\mathbf{v}(t) + \mathbf{B}_\mathrm{n}\mathbf{u}(t),\end{aligned} \tag{13}$$

where $\mathbf{u} = [u_v, u_\omega]$ are velocity inputs and

$$\mathbf{A}_\mathrm{n} = \begin{bmatrix} -\frac{1}{\tau_v} & 0 \\ 0 & -\frac{1}{\tau_\omega} \end{bmatrix}, \quad \mathbf{B}_\mathrm{n} = \begin{bmatrix} \frac{k_1}{\tau_v} & 0 \\ 0 & \frac{k_2}{\tau_\omega} \end{bmatrix}. \tag{14}$$

The simplified system is dynamically equivalent to (12). We identify the process gains k_1, k_2 and the process time constants τ_v, τ_ω using system identification on the real robot. In our case, the robot is symmetric and rotates around the origin, therefore x_ICR is assumed 0.

C. Adaptive Tracking Controller

First, we explain why using a control matrix adaptation is suitable for adapting to longitudinal and rotational slips, as well as the internal dynamics of a tracked vehicle. Next, we design a composite adaptive controller for the system in (13) and prove its exponential convergence to a bounded error ball. Note that our adaptive controller can be applied to any system of the form (7).

1) Motivation for Control Matrix Adaptation: The longitudinal slip κ is defined [82] as:

$$\kappa = -\frac{v_x^\mathcal{B} - \Omega_\mathrm{tr} r_\mathrm{tr}}{v_x^\mathcal{B}}, \tag{15}$$

where Ω_tr is the angular velocity of the tracks, r_tr is the track wheel radius, and $v_x^\mathcal{B}$ is the projection of the inertial velocity onto the body x-forward axis. Let $\Omega_\mathrm{tr} r_\mathrm{tr}$ be our velocity control input u_v. Then (15) can be written as $v_x^\mathcal{B} = \frac{1}{1+\kappa} u_v$. Analyzing the extreme cases, we notice that if $\kappa = 0$ (no longitudinal slip), the velocity of the vehicle will match the velocity input into the tracks. In comparison, if $\kappa \to \infty$ (entirely longitudinal slip), the forward velocity of the vehicle will tend toward zero. Similar reasoning can be applied to the rotational slip. Therefore, adapting for a coefficient that multiplies the control input (the track speeds) ensures tracking of the body's forward and angular velocity.

In addition, adapting the control matrix also contributes to compensating for the unknown internal dynamics of the robot, because the velocity control input is the setpoint to an internal proportional-derivative-integral controller, which outputs motor torques to the tracked vehicle. Lastly, adapting the control matrix effectively compensates for track degradation, manifested as a slowdown in the sprocket's angular velocity.

2) Reference Trajectories: We define a 2 dimensional *feasible* trajectory characterized by the desired position and velocity $\mathbf{p}_d^\mathcal{I}(t)$, $\mathbf{v}_d^\mathcal{I}(t)$ in the inertial frame $\mathcal{I}$. The position error is $\tilde{\mathbf{p}}^\mathcal{I} = \mathbf{p}^\mathcal{I} - \mathbf{p}_d^\mathcal{I}(t)$, where $\mathbf{p}^\mathcal{I} = [p_x^\mathcal{I}, p_y^\mathcal{I}]$, and ψ_d is the desired yaw angle. Let the following reference velocities be defined as:

$$\mathbf{v}_\mathrm{ref}^\mathcal{I} = \mathbf{v}_d^\mathcal{I} - \mathbf{K}_p \tilde{\mathbf{p}}^\mathcal{I}, \tag{16}$$

$$v_\mathrm{ref,x} = \begin{bmatrix} \cos(\psi) \\ \sin(\psi) \end{bmatrix} \cdot \mathbf{v}_\mathrm{ref}^\mathcal{I}, \tag{17}$$

$$\omega_\mathrm{ref} = \dot{\psi}_\mathrm{ref} - k_\psi(\psi - \psi_\mathrm{ref}), \tag{18}$$

where

$$\psi_\mathrm{ref} = \begin{cases} \arctan\left(\frac{\mathbf{v}_\mathrm{ref,y}^\mathcal{I}}{\mathbf{v}_\mathrm{ref,x}^\mathcal{I}}\right), & \text{if } \|\mathbf{v}_\mathrm{ref}^\mathcal{I}\|_2^2 > v_\epsilon \\ \psi_\mathrm{d}, & \text{otherwise.} \end{cases} \tag{19}$$

Note that the reference trajectory is not fully pre-planned; it includes feedback terms that are only defined during the execution of the trajectory. Both $\mathbf{K}_p \in \mathbb{R}^{2\times 2}$ and $k_\psi \in \mathbb{R}$ are positive gains, with $\mathbf{K}_p = \mathrm{diag}(k_{px}, k_{py})$, and v_ϵ is a small velocity constant used to ensure the robot can track time-varying position trajectories, as well as turn in place. We define $\mathbf{v}_\mathrm{ref} = [v_\mathrm{ref,x}, \omega_\mathrm{ref}]^\top$, which is our reference trajectory.

3) Controller Synthesis: We design a composite adaptive controller $\mathbf{u}(t)$ and show that this composite tracking error and adaptation error exponentially converges to a bounded error ball. The composite tracking error variable $\mathbf{s}$ is defined as:

$$\mathbf{s} = \mathbf{v} - \mathbf{v}_\mathrm{ref} = [v_x^\mathcal{B} - v_\mathrm{ref,x},\ \omega - \omega_\mathrm{ref}]^\top. \tag{20}$$

Now, the tracking controller becomes:

$$\mathbf{u} = -(\mathbf{B}_\mathrm{n} + \boldsymbol{\Phi}\hat{\boldsymbol{\theta}})^{-1}[\mathbf{K}\mathbf{s} + \mathbf{A}_\mathrm{n}\mathbf{v}_\mathrm{ref} - \dot{\mathbf{v}}_\mathrm{ref}], \tag{21}$$

where $\mathbf{K} \in \mathbb{R}^{2\times 2}$ is a positive gain matrix, with $\mathbf{K} = \mathrm{diag}(k_{dx}, k_{dw})$.

Corollary 1. *By applying the controller in (22) to the dynamics that evolve according to (13), with the composite adaptation law:*

$$\dot{\hat{\boldsymbol{\theta}}} = -\lambda\hat{\boldsymbol{\theta}} - \boldsymbol{\Gamma}\mathbf{H}^\top\mathbf{R}^{-1}(\mathbf{H}\hat{\boldsymbol{\theta}} - \mathbf{y}) + \boldsymbol{\Gamma}\mathbf{H}^\top\mathbf{s}, \tag{22a}$$

$$\dot{\boldsymbol{\Gamma}} = -2\lambda\boldsymbol{\Gamma} + \mathbf{Q} - \boldsymbol{\Gamma}\mathbf{H}^\top\mathbf{R}^{-1}\mathbf{H}\boldsymbol{\Gamma}, \tag{22b}$$

where $\lambda > 0$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{n_\theta \times n_\theta}$, $\mathbf{Q}^{n_\theta \times n_\theta}$, and $\mathbf{R} \in \mathbb{R}^{2\times 2}$ are positive definite matrices, then the tracking errors $\mathbf{s}$ and the parameter error $\tilde{\boldsymbol{\theta}}$ will exponentially converge to a bounded error ball defined in [27].

Proof. By observing that $\sum_{i=1}^{n_\theta} \boldsymbol{\Phi}_i \hat{\theta}_i \mathbf{u} = \mathbf{H}\hat{\boldsymbol{\theta}}$ as emphasized in (3), the proof of exponential convergence from [27] can be directly applicable. □

Note that the convergence proof for exponential convergence for Corollary 1 using the Lyapunov theory holds for both when the last term of the gain adaptation (22b) is positive and when the last term is negative. Under the Kalman Filter interpretation, a negative sign is more intuitive as it emphasizes that the covariance $\boldsymbol{\Gamma}$ decreases with more measurements. Next, for simplicity, we assume the adaptation gain law in (22b) does not have cross terms, and thus there is one positive adaptation gain γ_i for each $\hat{\theta}_i$, with $i \in [1, n_\theta]$. Under this assumption, we prove the exponential convergence of both $\tilde{\theta}_i$ and $\mathbf{s}$ to a bounded error ball. We also show that, under this assumption, the last term in (22b) needs to be positive in order to prove stability.

Theorem 1. *By applying the controller in (21) to the dynamics that evolve according to (13), with the composite adaptation law, for each $i \in [1, n_\theta]$:*

$$\dot{\hat{\theta}}_i = -\lambda\hat{\theta}_i - \gamma_i \mathbf{u}^\top \boldsymbol{\Phi}_i^\top \mathbf{R}^{-1}\left(\sum_{i=1}^{n_\theta} \boldsymbol{\Phi}_i \mathbf{u}\hat{\theta}_i - \mathbf{y}\right) + \gamma_i \mathbf{s}^\top \boldsymbol{\Phi}_i \mathbf{u},$$
$$\dot{\gamma}_i = -2\lambda\gamma_i + q_i + \gamma_i \mathbf{u}^\top \boldsymbol{\Phi}_i^\top \mathbf{R}^{-1}\boldsymbol{\Phi}_i \mathbf{u}\gamma_i, \tag{23}$$

where $\gamma_i > 0,\ q_i > 0$, for each $i \in [1, n_\theta]$, then the tracking errors $\mathbf{s}$ and the parameter error $\tilde{\boldsymbol{\theta}}$ will exponentially converge to a bounded error ball.

Proof. Defining the true control matrix as $\mathbf{B} := \mathbf{B}_\mathrm{n} + \boldsymbol{\Phi}\boldsymbol{\theta}$, we obtain the following closed loop system in (13):

$$\dot{\mathbf{v}} = \mathbf{A}_\mathrm{n}\mathbf{v} - (\mathbf{B}_\mathrm{n} + \boldsymbol{\Phi}\boldsymbol{\theta})(\mathbf{B}_\mathrm{n} + \boldsymbol{\Phi}\hat{\boldsymbol{\theta}})^{-1}[\mathbf{K}\mathbf{s} + \mathbf{A}_\mathrm{n}\mathbf{v}_\mathrm{ref} - \dot{\mathbf{v}}_\mathrm{ref}] + \boldsymbol{\delta}.$$

Let $\tilde{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}$ be the error adaptation vector. Further, using the composite variable $\mathbf{s}$, defined in (20), the closed-loop system becomes:

$$\dot{\mathbf{s}} = \mathbf{A}_\mathrm{n}\mathbf{s} - \mathbf{K}\mathbf{s} - (\boldsymbol{\Phi}\tilde{\boldsymbol{\theta}})\mathbf{u} = \mathbf{A}_\mathrm{n}\mathbf{s} - \mathbf{K}\mathbf{s} - \sum_{i=1}^{n_\theta} \boldsymbol{\Phi}_i \mathbf{u}\tilde{\theta}_i + \boldsymbol{\delta}.$$

For the prediction term in (23), we define the dynamics residual derivative $\mathbf{y}$ determined for the bounded and adversarial noise $\bar{\epsilon}$ as $\mathbf{y} = \mathrm{LPF(s)}\dot{\mathbf{x}} - \mathbf{f}(\mathbf{x}, \mathbf{u}, t) = (\boldsymbol{\Phi}\boldsymbol{\theta})\mathbf{u} + \bar{\epsilon}$, where premultiplying the noisy measurement $\dot{\mathbf{x}}$ by $\mathrm{LPF}(s)$ with the Laplace transform variable s indicates low-pass filtering. Using the Lyapunov function: $\mathcal{V} = \mathbf{s}^\top\mathbf{s} + \sum_{i=1}^{n_\theta} \tilde{\theta}\gamma_i^{-1}\tilde{\theta}$, we compute its derivative, as follows:

$$\begin{aligned}
\dot{\mathcal{V}} &= 2\mathbf{s}^\top\dot{\mathbf{s}} + 2\sum_{i=1}^{n_\theta} \tilde{\theta}_i\gamma_i^{-1}\dot{\tilde{\theta}}_i + \sum_{i=1}^{n_\theta} \tilde{\theta}_i \frac{d}{dt}\left(\gamma_i^{-1}\right)\tilde{\theta}_i \\
&= -2\mathbf{s}^\top\left[(\mathbf{K} - \mathbf{A}_\mathrm{n})\mathbf{s} + \sum_{i=1}^{n_\theta} \boldsymbol{\Phi}_i \mathbf{u}\tilde{\theta}_i\right] \\
&+ 2\sum_{i=1}^{n_\theta} \gamma_i^{-1}\tilde{\theta}_i(\gamma_i \mathbf{s}^\top \boldsymbol{\Phi}_i \mathbf{u} - \gamma_i \mathbf{u}^\top \boldsymbol{\Phi}_i^\top \mathbf{R}^{-1}\sum_{j=1}^{n_\theta} \boldsymbol{\Phi}_j \mathbf{u}\tilde{\theta}_j - \lambda\tilde{\theta}_i) \\
&+ \sum_{i=1}^{n_\theta} \tilde{\theta}_i(2\gamma_i^{-1}\lambda - \gamma_i^{-1}q_i\gamma_i^{-1} - \mathbf{u}^\top \boldsymbol{\Phi}_i^\top \mathbf{R}^{-1}\boldsymbol{\Phi}_i \mathbf{u})\tilde{\theta}_i \\
&\underbrace{+2\left(\mathbf{s}^\top\boldsymbol{\delta} + \sum_{i=1}^{n_\theta} \tilde{\theta}_i\left(\mathbf{u}^\top \boldsymbol{\Phi}_i^\top \mathbf{R}^{-1}\bar{\epsilon} - \gamma_i^{-1}\lambda\theta_i - \gamma_i^{-1}\dot{\theta}_i\right)\right)}_{\text{error terms}}.
\end{aligned}$$

After further manipulation, the time derivative of the Lyapunov function becomes:

$$\begin{aligned}
\dot{\mathcal{V}} &= -2\mathbf{s}^\top(\mathbf{K} - \mathbf{A}_\mathrm{n})\mathbf{s} - \sum_{i=1}^{n_\theta} \tilde{\theta}_i(q_i\gamma_i^{-2} + \mathbf{u}^\top \boldsymbol{\Phi}_i^\top \mathbf{R}^{-1}\boldsymbol{\Phi}_i \mathbf{u})\tilde{\theta}_i \\
&- 2\left(\sum_{i=1}^{n_\theta} \tilde{\theta}_i \mathbf{u}^\top \boldsymbol{\Phi}_i^\top\right)\mathbf{R}^{-1}\left(\sum_{j=1}^{n_\theta} \boldsymbol{\Phi}_j \mathbf{u}\tilde{\theta}_j\right) + \text{error terms}.
\end{aligned} \tag{24}$$

There exists $\alpha \in \mathbb{R}_+$ such that:

$$\begin{aligned}
-2(\mathbf{K} - \mathbf{A}_\mathrm{n}) &\preceq -2\alpha\mathbf{I}, \\
q_i\gamma_i^{-2} + \mathbf{u}^\top \boldsymbol{\Phi}_i^\top \mathbf{R}^{-1}\boldsymbol{\Phi}_i \mathbf{u} &\leq -2\alpha\gamma_i^{-1}, \quad \forall i \in [1, n_\theta].
\end{aligned} \tag{25}$$

We assume that $\|\boldsymbol{\delta}\|$, $\bar{\epsilon}$, and $\dot{\theta}_i$ are small and bounded, and that the true value θ_i is bounded. In addition, the DNN $\boldsymbol{\Phi}_i$ is bounded since we use spectral normalization and the input domain is bounded. We then define an upper bound for the error terms as:

$$\bar{d} = \sup_t\left(\|\boldsymbol{\delta}\| + \left|\sum_{i=1}^{n_\theta}\left(\mathbf{u}^\top \boldsymbol{\Phi}_i^\top \mathbf{R}^{-1}\bar{\epsilon} - \gamma_i^{-1}\lambda\theta_i - \gamma_i^{-1}\dot{\theta}_i\right)\right|\right), \tag{26}$$

and define the matrix $\mathcal{M}$, for $i \in [1, n_\theta]$:

$$\mathcal{M} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathrm{diag}(\gamma_i^{-1}) \end{bmatrix}. \tag{27}$$

By applying the Comparison Lemma [83], we can then prove the tracking error and adaptation parameters exponentially converges to the bounded error ball:

$$\lim_{t\to\infty}\left\|\begin{bmatrix} \mathbf{s} \\ \tilde{\theta}_i \end{bmatrix}\right\| \leq \frac{\bar{d}}{\alpha\lambda_{\min}(\mathcal{M})}, \tag{28}$$

where $\lambda_{\min}$ is the minimum eigenvalue of a square matrix.

Now it follows from [83], [84] that the input-to-state stability (ISS) and bounded input and bounded output (BIBO) stability in the sense of finite-gain $\mathcal{L}_p$ [83] is proven for $d \in \mathcal{L}_{pe}$, resulting in its bounded output $\mathbf{s}, \tilde{\boldsymbol{\theta}} \in \mathcal{L}_{pe}$, where the $\mathcal{L}_p$ norm in the extended space $\mathcal{L}_{pe}, p \in [1, \infty]$ is:

$$\begin{aligned}
\|(\mathbf{u})_\tau\|_{\mathcal{L}_p} &= \left(\int_0^\tau \|\mathbf{u}(t)\|^p dt\right)^{1/p} < \infty, \quad p \in [1, \infty) \\
\|(\mathbf{u})_\tau\|_{\mathcal{L}_\infty} &= \sup_{t\geq 0}\|(\mathbf{u}(t))_\tau\| < \infty
\end{aligned}$$

and $(\mathbf{u}(t))_\tau$ is a truncation of $\mathbf{u}(t)$, i.e., $(\mathbf{u}(t))_\tau = \mathbf{0}$ for $t \geq \tau$, $\tau \in [0, \infty)$ while $(\mathbf{u}(t))_\tau = \mathbf{u}(t)$ for $0 \leq t \leq \tau$.

□

In contrast with [27], [76], (21) and (23) admits adaptation through the $\mathbf{B}$ control influence matrix and for stability purposes, under the assumption of a diagonal $\boldsymbol{\Gamma}$, the adaptation law equation resembles the Ricatti equation of the $\mathcal{H}_\infty$ filtering [85]. This tends to increase the adaptation gain, making it more responsive to measurements.

The parameters of the adaptation law (23) are $\boldsymbol{\Gamma} = \mathrm{diag}(\gamma_1, \ldots, \gamma_{n_\theta})$, $\mathbf{R}$, λ, and $\mathbf{Q} = \mathrm{diag}(q_i)$. $\boldsymbol{\Gamma}$ is a positive definite matrix that influences the convergence rate of the estimator, and a sufficiently large initial $\boldsymbol{\Gamma}_0$ should be chosen to obtain a suitable convergence rate. $\mathbf{Q}$ is a positive definite gain added to the gain update law. λ is a damping factor. $\mathbf{R}$ is a gain added to the prediction component of the adaptation law. Without this gain, the prediction term and the tracking error-based term could not be tuned separately.

Note that the convergence proof for exponential convergence for Corollary 1 holds for both when the last term of the gain adaptation is positive and when the last term is negative. Under the Kalman Filter interpretation, a negative sign is more intuitive as it emphasizes that the covariance $\boldsymbol{\Gamma}$ decreases with more measurements.

Lastly, for completeness, we show exponential convergence to a bounded error ball for the position and the attitude error.

Theorem 2. *By Theorem 1,* $\mathbf{s}$ *converges to a bounded error ball* (28) *defined as* $\bar{b}$. *Therefore, we hierarchically show that* $\psi \to \psi_{\text{ref}}$ *and* $\mathbf{p} \to \mathbf{p}_d$ *exponentially fast to a bounded error ball for bounded reference velocity.*

Proof. We define the error $\tilde{\psi} = \psi - \psi_{\text{ref}}$. Using (18), we obtain $\dot{\tilde{\psi}} + k_\psi \tilde{\psi} \leq \bar{b}$, and with the Comparison Lemma, we prove that the error $\tilde{\psi}$ converges to the bounded error ball $\frac{\bar{b}}{k_\psi}$. To give intuition about the following position tracking error proof, we use Fig. 4. We define in vector form $\mathbf{v} = \mathbf{v}_{\text{ref}} + \mathbf{v}_e$, where $\mathbf{v}_e$ is the velocity error. We further express these quantities in the reference frame $\mathcal{D}$ and note that, by Theorem 1, we have proved the convergence $v_x^{\mathcal{B}} = v_{\text{ref,x}} + \bar{b}$ as $t \to \infty$. Therefore, we obtain:

$$\mathbf{v}_e^{\mathcal{D}} = -\begin{bmatrix} v_{\text{ref,x}}^{\mathcal{D}} \\ 0 \end{bmatrix} + \begin{bmatrix} \cos(\tilde{\psi}) \\ \sin(\tilde{\psi}) \end{bmatrix} \left(\begin{bmatrix} \cos(\tilde{\psi}) \\ \sin(\tilde{\psi}) \end{bmatrix} \cdot \begin{bmatrix} v_{\text{ref,x}}^{\mathcal{D}} \\ 0 \end{bmatrix} + \bar{b} \right)$$
$$= v_{\text{ref,x}}^{\mathcal{D}} \begin{bmatrix} \cos^2(\tilde{\psi}) - 1 \\ \sin(\tilde{\psi})\cos(\tilde{\psi}) \end{bmatrix} + \bar{b} \begin{bmatrix} \cos(\tilde{\psi}) \\ \sin(\tilde{\psi}) \end{bmatrix}.$$

We compute and bound the norm, as follows:

$$\|\mathbf{v}_e\|_2 \leq v_{\max} \left| \sin\left(\frac{\bar{b}}{k_\psi} \right) \right| \sqrt{2} + \bar{b}, \qquad (29)$$

where $v_{\max}$ is our assumption for the existence of an upper bound for the reference velocity. From (16) and (29), it is straightforward to see that the position error is also bounded. □

V. Empirical Results: Analysis of VFM Suitability

For our empirical work, we selected Dino V1 [86] as the VFM. Dino maps a high-resolution red-green-blue (RGB) image to a lower-resolution image where each pixel is a high-dimensional feature vector that depends on the *entire* input image, not just the corresponding input patch. More precisely, let $\xi \in \mathbb{N}$ be the patch dimension and $\xi_f \in \mathbb{N}$ the feature vector dimension. Given an RGB image $\mathcal{I}_{\text{RGB}}$, the transformation is:

$$\mathcal{I}_{\text{RGB}} : h \times w \times 3 \to \mathcal{I}_{\text{VFM}} : \left\lfloor \frac{h}{\xi} \right\rfloor \times \left\lfloor \frac{w}{\xi} \right\rfloor \times \xi_f, \qquad (30)$$

where h is the image height and w is the image width. We then extract prominent patch(es) from $\mathcal{I}_{\text{VFM}}$ to form the terrain representation $\boldsymbol{\mathcal{E}}$, which will be further used in the $\mathbf{\Phi}$ from (3).

Dino is optimized for a self-supervised learning objective and was shown to yield feature mappings useful for a variety of downstream tasks. This VFM is trained on the Google Landmarks v2 dataset [87], which includes diverse ground terrains but mainly in the context of buildings, plants, landscapes, etc., instead of terrain-only images. Therefore, in this section, we verify that Dino is able to clearly discriminate between different terrain types in terrain-only images before deploying it in our control setting.

We first measure Dino's discriminative ability by examining the margins of linear classifiers between terrain classes in the high-dimensional feature space $\mathbb{R}^{\xi_f}$. We consider three terrain types: grass, sand, and snow. We collect five example images for each class and convert each image to a set of feature vectors using Dino. Then, using the known class labels, we fit a multi-class linear classifier for the feature vectors using the One-vs-Rest Support Vector Classifier (OVR-SVC) method [88]. We then project the feature vectors onto the one-vs-rest separating hyperplane normals. Let the separating hyperplane have the equation $\mathbf{w}_h \cdot \mathbf{x} + \mathbf{b}_h = 0$, where $\mathbf{w}_h \in \mathbb{R}^{\xi_f}$ is the vector normal to hyperplane and $\mathbf{b}_h \in \mathbb{R}$ is the bias term. Let $\boldsymbol{\mathcal{E}}$ be represented by just one patch, and thus have size ξ_f. The projected patch $\boldsymbol{\mathcal{E}}$ onto the separating hyperplane normal is defined as $p_h = \mathbf{w}_h \cdot \boldsymbol{\mathcal{E}}$. The histogram of these projected values for each patch in the image is shown in Fig. 5. By comparing the SVC margin (the separation between -1 and 1) to the width of the histograms, we confirm that the classes are highly separable. (Note that the spikes at -1 and 1 are an artifact of the high dimensionality and the small dataset we used.)

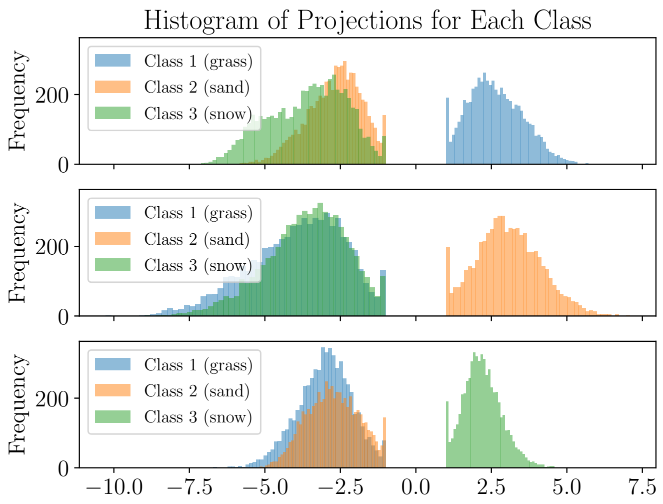


Fig. 5: Dino VFM discriminative ability for different terrains. We show the histograms of the projection values onto the separating hyperplane normal computed using Support Vector Classifier for 3 sets of classes with 5 images each (each row presents the separation margin between one class type and the other 2 classes).

We next examine the distribution of the features across a sequence of images, taken while navigating from flagstone (irregular-shaped flat rocks) to gravel in the Mars Yard [89] at NASA Jet Propulsion Laboratory (JPL). The top row of Fig. 6 displays 8 out of a total of 45 images extracted from a video. Each image is processed through the Dino VFM, yielding 1200 patches of dimension $\xi_f = 384$ per image (computed using (30)). We apply OVR-SVC on the patches from one flagstone and one gravel image and project all patches from our chosen 8 images onto the SVC separating hyperplane normal. This projection reveals a bimodal distribution in the 3rd and 4th images due to the presence of both flagstones and gravel. In the bottom subplot of Fig. 6, we simulate a scenario where the robot traverses the area covered in all 45 images sequentially. For each image, we focus on a central patch of size ξ_f, and project these features onto the separating hyperplane normal. This projection shows a consistent and continuous trend as the robot transitions from flagstone to

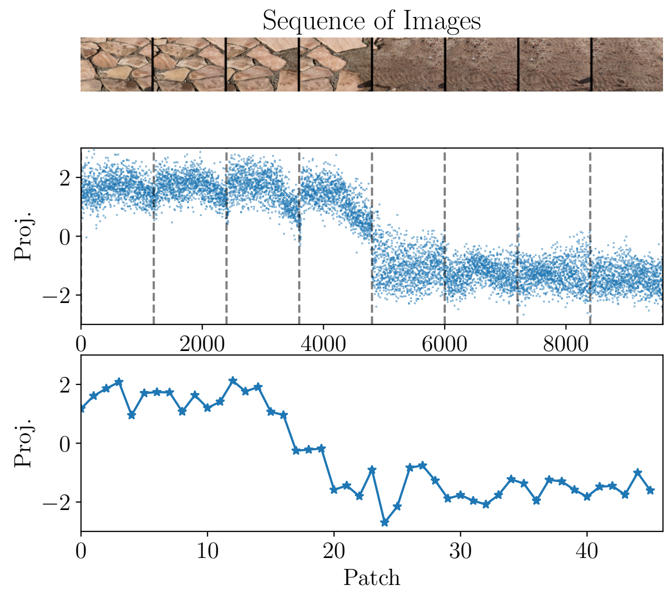


Fig. 6: Projection of sequential flagstone and gravel features onto an OVR-SVC separating hyperplane normal. The middle plot shows the projection of all the patch features from the top 8 figures, while the bottom plot shows the projection of a central patch taken from 45 sequential images of flagstone and gravel.

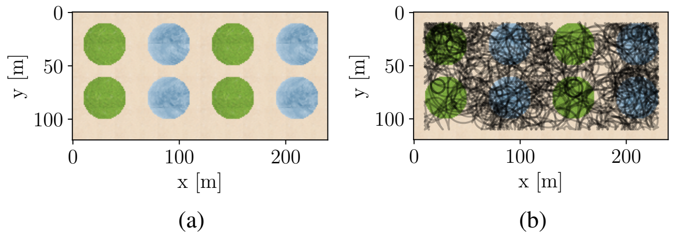


Fig. 7: (a) Environment with 3 different types of terrain (sand, grass, and ice), which represent areas of differing slip factors (b) Generated trajectories for training.

gravel surfaces. This observation ensures the continuity of the VFM with respect to the camera motion.

Overall, these results provide positive empirical evidence that the Dino VFM is suitable for fine-grained discrimination of terrain types in images containing only terrain, and thus suitable for use in our setting.

VI. Empirical Results: Simulation Studies

A. Simulation Study Settings

To validate our learning and control strategy, we developed a simulation environment (Fig. 7) that enables detailed visualizations of the algorithm behavior. The dynamics for the simulator were modeled via (13), and the controller of (21), (23) with the coefficients in Table I was used to track user-defined velocity trajectories $\mathbf{v}_{\text{ref}}$ generated at random. The environment contains three distinct terrain types (Fig. 7a). Each terrain type induces a different level of slip, modeled as a scaling of the matrix $\mathbf{B}_n$ in (13) such that $\mathbf{B}_n$ is replaced by $\eta\mathbf{B}_n$. The particular values of η are illustrated in Fig. 8a.

To construct a dataset, we simulate $N = 1$ long trajectory of 150 000 discrete time steps, with randomized piecewise-constant velocity inputs. For acquiring these features, we utilize the Dino VFM on images of the terrain. As explained in Section V, this model processes high-resolution terrain images into a more compact, lower resolution embedding. This reduced resolution representation is overlayed across the entire map. Specifically, let $m_s \times n_s$ be the size of the simulated map, which is 120×240 in our case. Let each Dino feature image have the size computed as in (30) (in our case, $30 \times 40 \times \xi_f$). Thus, we tile each of the Dino feature images across the entire map and extract and record the relevant terrain features underneath the robot. For training, we collect random trajectories, generated by sampling control inputs $\mathbf{u}$ from a uniform distribution, and integrate forward the dynamics in (13), in order to cover a large portion of the simulated map, as seen in Fig. 7b. The dataset contains the Dino features extracted from underneath the robot and the robot's velocities, and as labels the residual dynamics derivative $\mathbf{y}$. Using this dataset, we then train the basis function $\mathbf{\Phi}$, whose architecture can be seen in Fig. 3, using Algorithm 1. The convergence of the adaptation coefficients to $\boldsymbol{\theta}_r$ is presented in Fig. 8b, taken as an average over the last training minibatch K. This compact representation of the terrain is then integrated with online adaptive control (Algorithm 2).

TABLE I: Control and adaptation coefficients for the $\mathbf{\Phi}$ constant and $\mathbf{\Phi}$ DNN controllers for the simulation environment

Ctrl. Type	k_{dx}	$k_{d\omega}$	$\mathbf{\Gamma}_0$ diag	$\mathbf{R}$ diag
$\mathbf{\Phi}$ = ct.	0.05	0.1	0.001	0.002
$\mathbf{\Phi}$ = NN	0.05	0.1	0.001	0.01

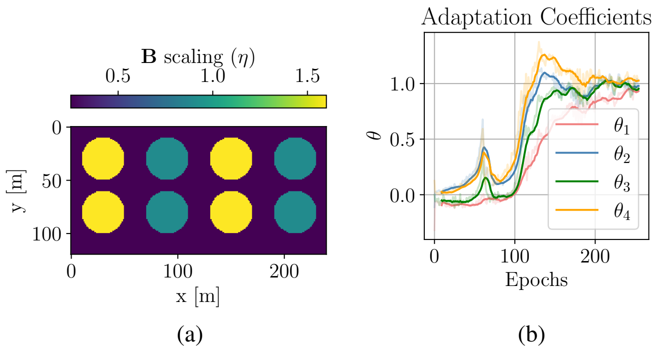


Fig. 8: (a) Perturbed control matrix (η) on the different types of terrains. (b) Convergence of the adaptation coefficients for the simulation dataset.

B. Simulation Study Objectives

In exploring the capabilities of our model, we investigate how prior knowledge of the terrain contribute to improved tracking accuracy for an adaptive controller. We thus test our algorithm across 3 scenarios: a) We assess the model performance in an environment identical to the one used during training to understand its effectiveness with in-distribution data (Fig. 9). b) We test the algorithm under simulated nighttime conditions to gauge performance when the ground is identical, but the lighting conditions are different (Fig. 10). c) We challenge the model by presenting it with two environments that have similar visual features to those in the training data

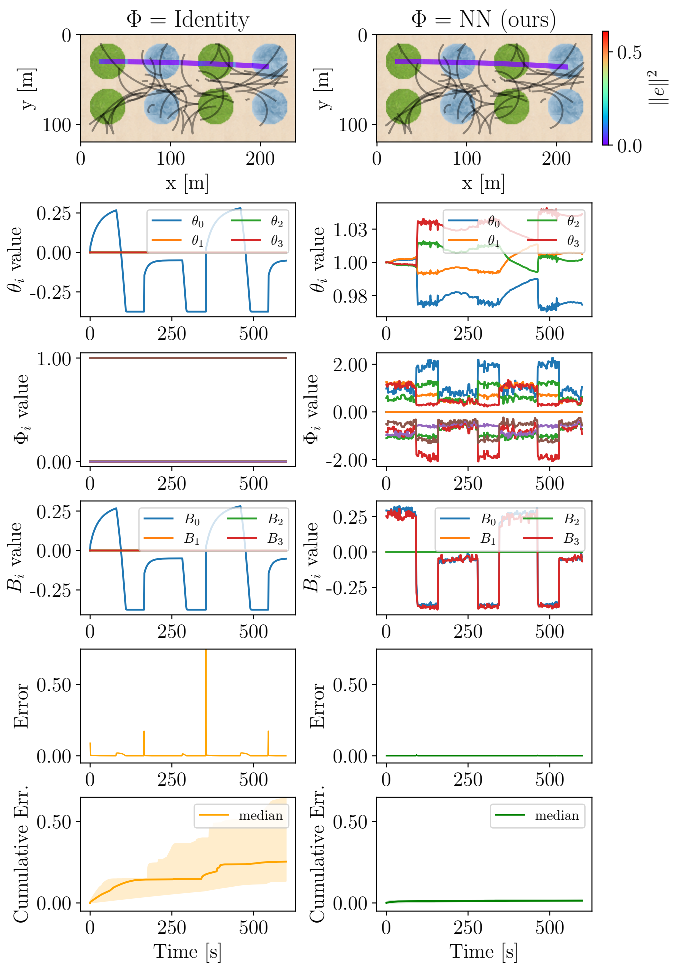


Fig. 9: Results for in-distribution data on the simulation model. Each experiment was run 40 times. The left column contains the performance for the baseline controller, while the right column contains the performance for our method. The last row presents the cumulative error, where the thick colored line represents the median, and the shaded region encompasses the range from the 25th to the 75th percentile.

set, but exhibit different dynamic behaviors. Furthermore, we adopt an adversarial approach by exposing the robot to completely novel environments that are not encountered during training (Fig. 12).

C. In-distribution Performance

To quantify if prior knowledge of the terrain improves tracking accuracy, the robot is tested in-distribution using the same environment as in the training dataset. The first row of Fig. 9 shows 39 random trials (black) and the single exemplar path (primarily purple, colored by the $\mathcal{L}_2$ error between the actual and desired states). These random trials are used to compute error statistics in row 6. The second row displays the robot's adaptation coefficients for the purple trajectory as it navigates through this environment. When the basis function lacks terrain awareness and is chosen constant (31), there is significant fluctuation in the adaptation coefficients during the transition between different terrains. Conversely, when the DNN basis function is used, the adaptation coefficients

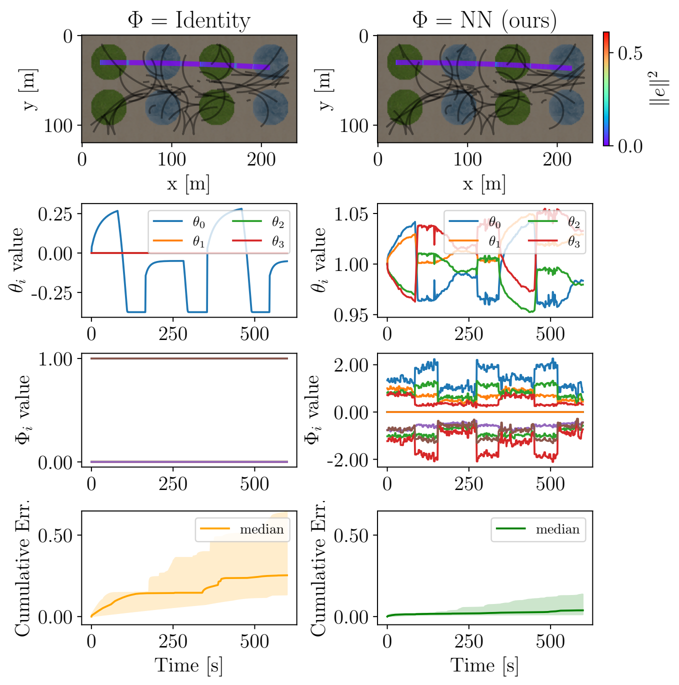


Fig. 10: Simulation results in the simulated nighttime environment. Despite different lighting conditions, the cumulative error is kept small by the terrain-informed DNN.

remain relatively stable, while the DNN output itself varies with each terrain type, as expected. The fourth row showcases the estimated $\mathbf{B}$ matrix, which is the product between the DNN basis function and the adaptation vector. Though the output of our controller is slightly noisier, the adaptation of the estimated $\mathbf{B}$ is significantly faster. The fifth row shows the normalized error between the robot's actual and desired states. For the constant basis function, most of the tracking error occurs at terrain transitions. When the basis function is terrain-informed, the error is insignificant, even at terrain transitions. Finally, the last row shows the spread of the cumulative error across 40 distinct experimental runs, each initiated at a random starting point and orientation, but of same duration (the black and the purple trajectories in Row 1). The results of the simulation show that our terrain-informed DNN-based tracking controller reduces the cumulative error by approximately 90.1% when compared to the constant $\mathbf{\Phi}$ define as in (31).

D. Nighttime Out-of-distribution Performance

To test the robustness of our framework to varying lighting conditions, we extend our simulated experiments with a nighttime environment by uniformly darkening (changing the brightness) of each image representing the environment (Fig. 10). While the adaptation coefficients exhibit more variation compared to those in the standard, in-distribution scenario, the DNN still demonstrates good accuracy in predicting the environment from the darkened images. This outcome emphasizes the robustness of VFMs, underscoring their ability to adapt effectively to varying lighting conditions. Importantly, even in these altered night conditions, the cumulative error remains low.

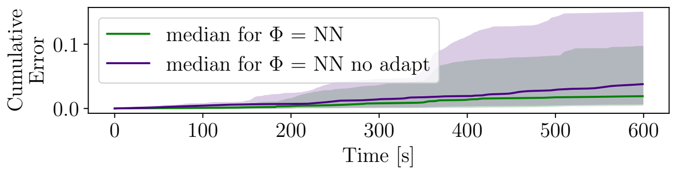


Fig. 11: Comparison of the cumulative error between adaptation and "no adaptation" for the simulated nighttime experiment. In both cases, the DNN version of $\mathbf{\Phi}$ is used.

In Fig. 11, we highlight the importance of adaptation by comparing the tracking error under two scenarios: with adaptation and without adaptation. In the "no adaptation case," we maintain $\boldsymbol{\theta}$ as a constant, initialized to $\boldsymbol{\theta}_r$. This comparison effectively demonstrates the benefits of adaptation, emphasizing its value even in situations where the basis function accurately predicts the terrain.

E. Adversarial Environment Performance

In our final test (Fig. 12), we introduced two adversarial environments for the robot, manipulating two visually similar environments by altering their respective η coefficient of the $\mathbf{B}$ matrix. This emulates the real world where pits of deep sand appear very similar to shallow sand, but have a significantly different effect on the dynamics of the robot. Additionally, we modified the appearance of the simulated ice environment to create a distinct visual difference, while also slightly changing the effect of ice on the dynamics.

In the adversarial environment, the adaptation coefficients exhibit greater changes than for the in-distribution and nighttime simulations. In addition, we observe that the DNN basis function demonstrates good performance, validating its effectiveness in handling out-of-distribution data. This effectiveness is likely attributed to the zero-shot capability inherent in the VFM. Lastly, it is important to note that the overall cumulative error remained lower compared to scenarios where the basis function lacked terrain information, further demonstrating the benefit and robustness of our approach in varied and challenging conditions, even for out-of-distribution data.

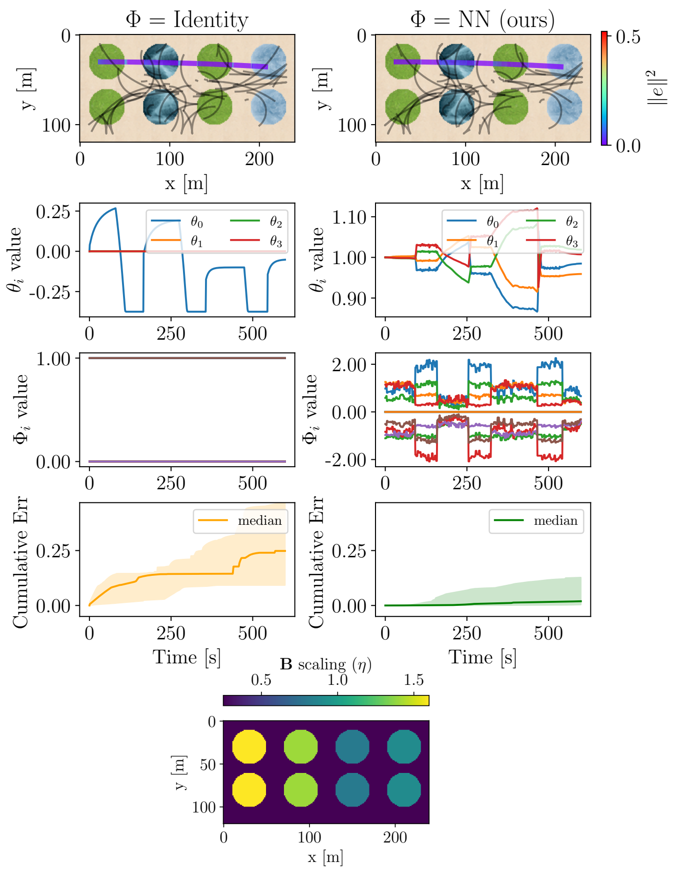


Fig. 12: Simulation results with adversarial input (different ice-looking structure) and different η coefficient for the $\mathbf{B}$ matrices for same-looking terrain.

VII. Empirical Results: Hardware Experiments

In this section, we focus on the hardware implementation and experimental validation of our MAGIC$^{\text{VFM}}$ adaptive controller discussed in Section III, V, and IV on a tracked vehicle whose dynamics are modeled in (13). We elucidate how our adaptive controller effectively addresses various perturbations such as terrain changes, severe track degradation, and unknown internal robot dynamics.

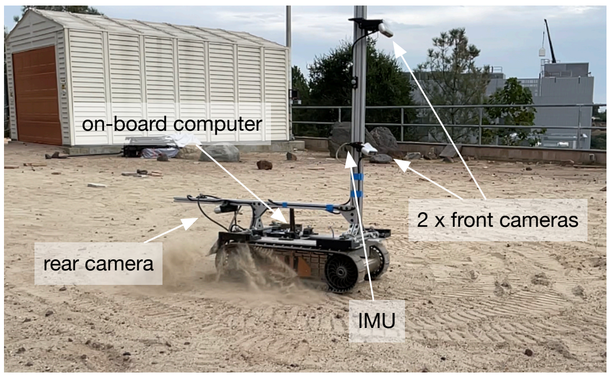


Fig. 13: The GVR-Bot with the sensing and compute units highlighted. Note that the forward facing camera is used for state estimation, while the top camera is used for taking terrain images for MAGIC$^{\text{VFM}}$. The rear camera is not used in this work.

A. Robot Hardware and Software Stack

Experiments were carried out using a GVR-Bot [90] (Fig. 13) equipped with an NVIDIA Jetson Orin, three RealSense D457 cameras (two forward facing and one rear facing) and a VectorNav VN100 IMU. Additionally, as the GVR-Bot does not have spring-loaded track tensioners, we installed an adjustable mount to passively align the tracks as the robot moves. This modification ensures the tracks do not fall off due to shear stress while traversing inclined grounds.

State estimation is provided onboard using OpenVINS [91], which fuses the camera data with an inertial measurement unit (IMU) to estimate the platform's position, attitude, and velocity. Our MAGIC$^{\text{VFM}}$ controller, as presented in (21), (23) and Theorem 1, is running at 20 Hz and it is implemented in Python using the Robot Operating System (ROS) as the middleware to communicate with the GVR-Bot's internal

TABLE II: Training hyperparameters for Algorithm 1. β is the learning rate for Line 15 of Algorithm 1, $\boldsymbol{\theta}_r$ is the regularization target, $\ell_{\min}$ and $\ell_{\max}$ are the bounds for the distribution over trajectory window lengths, λ_r is the regularization term for (4), K is the minibatch size, and n_θ is the size of the adaptation vector.

β	$\boldsymbol{\theta}_r$	$\ell_{\min}$	$\ell_{\max}$	λ_r	K	n_θ
0.001	$\mathbf{1}_4$	1.2 [s]	30 [s]	0.1	70	4

computer. The GVR-Bot accepts only velocity commands and the track velocities are regulated internally on the GVR-Bot using a PID motor controller that is inaccessible to the user. Due to this feature, we opt to model the robot's velocity as a first-order time delay system (13), with the parameters identified through system identification.

B. Experiments on Slopes in JPL's Mars Yard

To verify the performance of our MAGIC$^{\text{VFM}}$ controller (Section III and IV), the GVR-Bot was driven on the slopes of the Mars Yard [89] at the Jet Propulsion Laboratory (JPL). Fig. 14 shows the two selected slopes, both chosen for their appropriate angle and visually different terrain type that induce different dynamic terrain-based behaviors.

1) Offline Training: Training data was collected by driving the GVR-Bot via direct tele-operation for a total of 20 minutes on the slopes. This trajectory was designed to include segments of transition between different slopes as well as periods of single slope operation. We utilize this dataset for training our terrain-dependent basis function as outlined in Algorithm 1. By leveraging the strengths of a pre-trained VFM, we develop the lightweight DNN basis function head used in the MAGIC$^{\text{VFM}}$ adaptive controller of (21), (23). This function processes inputs comprising of the mean of two visual feature patches from the GVR-Bot's right and left tracks (resulting in 384 elements, see Fig. 2) as well as the robot's velocity taken from the onboard state estimator. The VFM-based DNN ($\boldsymbol{\Phi}$) structure incorporates two hidden layers, each consisting of 200 neurons, as seen in Fig. 3. The output has size 16, which is then reconfigured into dimensions $n \times m \times n_\theta$, where $n_\theta = 4$ represents the size of the adaptation vector, $m = 2$ is the size of the control input, and $n = 2$ denotes the state size. The hyperparameters for the training algorithm are shown in Table II.

2) Online Adaptation: At runtime, the downward-facing camera[2] is used to capture images of the terrain at 20 Hz. These images are then processed by the VFM explained in Section V to extract the features. The extracted features are then concatenated with the robot's velocity and are then fed into the DNN basis function $\boldsymbol{\Phi}$. This function, together with an online-adapting vector, is then employed to dynamically adjust the residual $\mathbf{B}$ matrix (21), (23) in real time to account for the different terrains.

The benefits of the terrain-informed basis function approach can been seen by comparing the performance of a constant basis function and non-constant basis function controller as the robot traverses slopes. Both controllers are based on (21) and the adaptation law in (23). The first controller uses a constant basis function, defined in (31), while the second controller uses a terrain-dependent DNN basis function trained as explained in Section VII-B1.

[2]To mitigate the purple tint in the RGB images (a common issue for Intel Realsense cameras), the RGB cameras were outfitted with neutral density filters. These filters are crucial in maintaining the integrity of the features, helping to prevent the input data from being skewed by abnormal coloration.

TABLE III: Control coefficients for both controllers ($\boldsymbol{\Phi}$ is constant and $\boldsymbol{\Phi}$ is a DNN).

k_{px}	k_{py}	k_{dx}	$k_{d\omega}$	k_{ψ}	$\boldsymbol{\Gamma}_0$ diag	$\mathbf{Q}$ diag	$\mathbf{R}$ diag	λ
0.8	0.8	0.5	1.6	2.3	0.2	0.1	5.0	0.01

TABLE IV: Tracking error statistics on Mars Yard slopes.

Controller	Tracking error [RMS/m]
$\mathbf{B}$ mat. adapt: $\boldsymbol{\Phi}$ = constant	0.130 ± 0.038
$\mathbf{B}$ mat. adapt: $\boldsymbol{\Phi}$ = DNN (ours)	0.061 ± 0.022

$$\boldsymbol{\Phi} = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right\}. \tag{31}$$

We choose this structure for the constant $\boldsymbol{\Phi}$ to capture both the direct and cross-term effects on the robot's velocity. The control coefficients for both controllers are presented in Table III. The initial adaptation vector $\boldsymbol{\theta}_0 = \boldsymbol{\theta}(0)$ for the constant basis function is $\mathbf{0}_{n_\theta}$, while $\boldsymbol{\theta}(0)$ for the terrain-dependent basis function is the converged value from Algorithm 1.

Each experiment was carried out five times, with the results detailed in Fig. 15. For repeatability, we used a rake to redistribute the gravel on the slopes between runs and alternate back and forth between the two controllers. For this experiment, the desired trajectory is a straight line that spans the entire length of the two slopes (see Fig. 14).

Fig. 15 shows that when the robot traverses the first slope (flagstone resulting in minimal slippage), both controllers have comparable tracking errors. However, a notable change in performance appears when the robot transitions to the second slope, which has an increased tendency for the soil to slump down the hill, causing slippage. In Table IV, we present the root-mean-square (RMS) error between the actual position and the desired position computed as $\sqrt{\frac{1}{L}\sum_{i=1}^{L} ||\mathbf{p}_i^{\mathcal{I}} - \mathbf{p}_{di}^{\mathcal{I}}||_2^2}$, where L is the length of the trajectory. The results demonstrate that the integration of a VFM in an adaptive control framework enhances tracking performance, yielding an average improvement of 53%.

3) Computational Load: A significant bottleneck in deploying visual foundation models onboard robots is the computational requirements of the inference stage of the models, especially as typical controllers need to run at 10s-100s Hz. To minimize the inference time and allow high controller rates, we employ the smallest visual transformer architecture of the Dino V1, consisting of 21M parameters. This architecture allows us to run the controller at 20 Hz on the Graphics Processing Unit (GPU).

C. Indoor Track Degradation Experiments

Indoor experiments were conducted at Caltech's Center for Autonomous Systems and Technologies (CAST) (Fig. 16). The primary objective of these experiments was to evaluate the robustness and performance of our proposed controller under

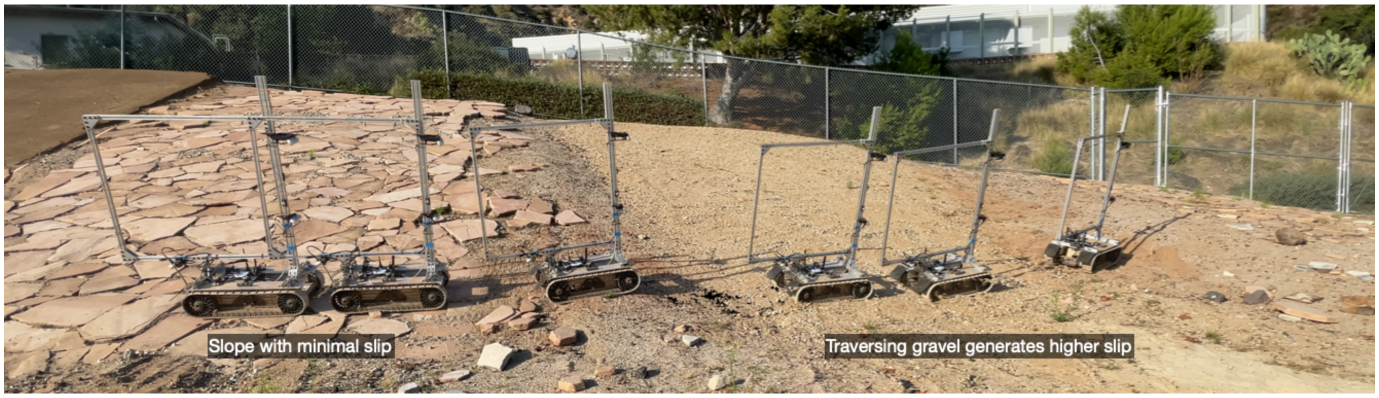


Fig. 14: The GVR-Bot traversing two slopes with different textures and terrain-induced dynamic behaviors at the JPL Mars Yard.

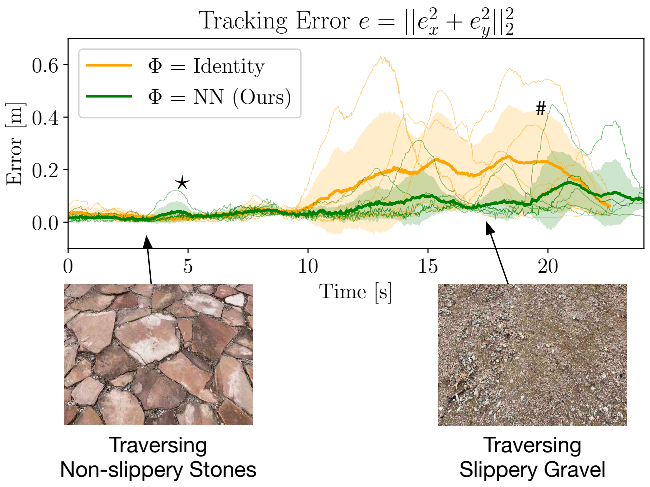


Fig. 15: Tracking error for the two controllers (constant basis function and terrain-dependent basis function) on the slopes. The error is computed as the Euclidean distance between actual and desired positions in the $\mathcal{I}$ frame. For both colors, the thick line outlines the mean of the 5 experiments, the shaded area 1 standard deviation, and the thin and transparent lines denote the 5 experiments. In ($\star$), our controller slipped more than expected due to loose gravel trapped between the flagstones, and in (#), the abrupt drop (higher than in the other experiments) from the slip caused a delay in recovery.

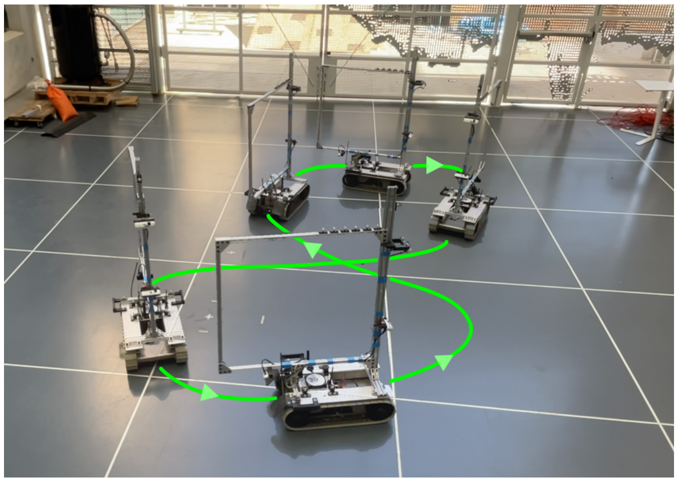


Fig. 16: Figure 8s for evaluating track degradation performance conducted indoors at CAST. The consistent floor of CAST ensures any slippage is consistent both within the figure 8 and between tests.

TABLE V: Control coefficients for the three controllers during the track degradation experiment

Ctrl. Type	k_{px}	k_{py}	k_{dx}	$k_{d\omega}$	k_{ψ}	$\mathbf{\Gamma}$ (diag)	$\mathbf{Q}$ (diag)	$\mathbf{R}$ (diag)	λ
nonlin. PD	0.8	0.8	0.5	1.6	2.3	-	-	-	-
$\mathbf{\Phi}$ ct.	0.8	0.8	0.5	2.3	2.0	1.0	0.1	5.0	0.001
$\mathbf{\Phi}$ NN	0.8	0.8	0.5	2.3	1.0	6.0	0.1	5.0	0.002

conditions of artificially-induced track degradation. Specifically, the experiments quantify the extent of degradation that our controller can effectively manage, and assess its advantage over baseline controllers in similar scenarios. We compare three controllers: (a) nonlinear PD ((21) without the adaptation), (b) MAGIC with constant $\mathbf{\Phi}$ in (21), (23), and (c) MAGIC$^{\text{VFM}}$ with DNN $\mathbf{\Phi}$ in (21), (23). The control coefficients for the three controllers are presented in Table V. Note that in this case, the DNN is not retrained on the new ground, but rather the previously trained DNN from Section VII-B is employed.

To simulate track degradation, a scalar factor is applied to one track that reduces its commanded rotation speed downstream of (and opaquely to) the controller. In this case, we apply a 70% reduction in speed to the right track using a step function with a period of 3 seconds, while keeping the left track operating 'nominally.' The GVR-Bot is commanded to follow a figure 8 trajectory, and the results are shown in Fig. 17 with the RMS errors in position tabulated in Table VI. The results show that both constant $\mathbf{\Phi}$ and DNN $\mathbf{\Phi}$ controllers outperform the tracking of the baseline PD controller by 22.6%, proving the robustness to model mismatch of both DNN and non-DNN controllers.

D. Performance at DARPA's Learning Introspection Control

The DARPA Learning Introspective Control (LINC) program [92] aims to develop machine learning-based introspection technologies that enable physical systems to respond to plant changes not predicted at design time, specifically with application to ground vehicles, ships, drone swarms, and robotic systems. LINC took place throughout 2023 at Sandia

TABLE VI: Position tracking error statistics with track degradation.

Controller	Tracking error (RMS/m)
PD	0.102
$\mathbf{B}$ Matrix Adapt. $\mathbf{\Phi}$ = constant	0.079
$\mathbf{B}$ Matrix Adapt. $\mathbf{\Phi}$ = NN.	0.070

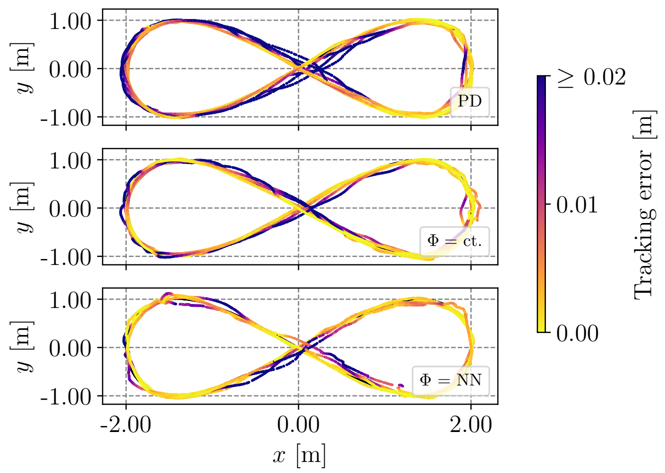


Fig. 17: Tracking error for the three controllers during track degradation. The first plot shows the tracking performance when running the nonlinear PD (the baseline). For the second and the third plot, we employ the constant basis function, and the terrain-informed basis function, respectively.

National Laboratories, culminating in the final evaluation exercise in December 2023.

The main exercise, Combined Circuit (Fig. 18), accessed accurate trajectory following under track degradation, safety (no collisions or tipping over), and reduced cognitive load on the driver across a variety of test elements. Importantly, these exercises were completed with a human driver as the global planner, introducing additional challenges such as adversarial driving, driver intent interfacing, and driver-induced oscillations not present in typical robotics projects that use high levels of autonomy.

For this exercise, we implemented the MAGIC controller from (21), (23) in which the basis function was (31) constant. Trajectories (both position and velocity) were generated using a sampling-based motion planner based on Monte Carlo Tree Search (MCTS) [93], with the desired goal locations generated using a 'driver intent' module that forecast a desired path based on operator joystick inputs. The model (13) used for the MAGIC controller was also employed by the MCTS-based planner and the estimated adaptation parameters $\hat{\boldsymbol{\theta}}$ were passed to the planner in real time, thus ensuring planning was being performed with the most up-to-date model. The main modules of the software stack and their interfaces are shown in Fig. 19, with our MAGIC controller highlighted in blue.

To evaluate the performance of our MAGIC controller, we compare the estimated state (linear velocity and angular velocity) from the visual-inertial odometry (VIO) with the reference trajectory $\mathbf{v}_{\text{ref}}$ computed from the desired trajectories generated by the MCTS-based planner (as explained in Section IV-C). For the baseline, we compare the desired command from the joystick with the actual state from the

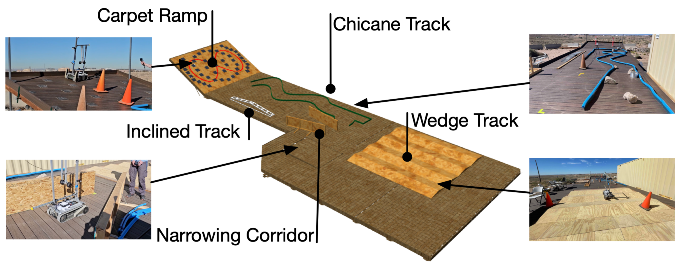


Fig. 18: Combined Circuit for the DARPA Learning Introspective Control test runs. The 3D render shows the full course set up with break outs of each of the elements. Competition track credit: Sandia National Laboratories team.

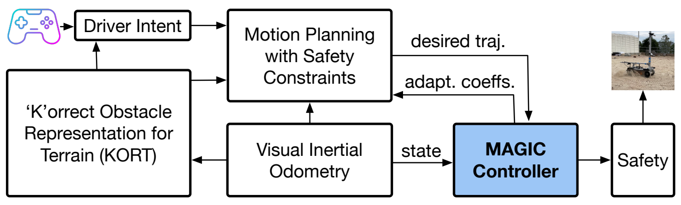

Driver Intent
Motion Planning with Safety Constraints
desired traj.
adapt. coeffs.
'K'orrect Obstacle Representation for Terrain (KORT)
Visual Inertial Odometry
state
MAGIC Controller
Safety

Fig. 19: Architecture for our DARPA Learning Introspective Control (LINC) software stack. In blue, we showcase our MAGIC controller, together with its interfaces to the rest of the system.

VIO. For both datasets, the GVR-Bot runs a fast proportional-integral-derivative (PID) controller on the error between the input velocity command and the wheel odometry.

The following subsections will discuss each of the components of the Combined Circuit and how our controller performs on each of them. In Table VII, we present the performance metrics for the four exercises of the LINC project. A particular exercise was traversed 4 times and the root-mean-square-error of the linear and angular velocity was computed.

1) Chicane Track: The Chicane Track highlighted the rejection of artificially induced track degradation, which was applied dynamically and opaquely as the GVR-Bot traversed the course. Due to the narrow track (the width is 0.9 m on average, 0.25 m wider than the GVR-Bot on both sides), track degradation leads to an increased number of collisions with the chicane walls if not quickly adapted to. Our MAGIC controller was able to successfully counteract and adapt to these challenges, thus making this artificially induced track degradation almost imperceptible to the driver after a very short initial adaptation transient.

The effectiveness of the trajectory tracking on the Chicane Track is shown in Fig. 20 for both the baseline and the MAGIC controller. In the first two rows, the tracking of the velocities is emphasized. The third row shows the amount of degradation applied to the system. The bottom plot shows the estimated adaptation parameters $\hat{\boldsymbol{\theta}}$ changing in real time to compensate for the track degradation. In Table VII, we report the improved performance of the MAGIC controller on this exercise. For the forward velocity, our controller improved tracking by 42.86%, while in angular velocity, by 20.45%. Because the track degradation information, $\mathbf{B}_n + \mathbf{\Phi}\hat{\boldsymbol{\theta}}$ of (21), is estimated by the MAGIC controller in real-time, the motion planner can successfully generate trajectories that makes use of this corrected control matrix, thereby successfully avoiding

TABLE VII: Performance metrics for the four exercises of the DARPA Learning Introspective Control (LINC) project

	Chicane			Carpet Ramp			Wedges			Narrow Corridor		
	v error [m/s]	ω error [rad/s]	Time [s]	v error [m/s]	ω error [rad/s]	Time [s]	v error [m/s]	ω error [rad/s]	Time [s]	v error [m/s]	ω error [rad/s]	Time [s]
LINC off	0.28	0.44	16.36	0.855	0.58	19.96	0.36	0.57	34.95	0.51	0.71	17.95
LINC on	0.16	0.35	44.36	0.36	0.31	39.22	0.24	0.33	39.72	0.18	0.43	23.73
Improvement	**42.9%**	**20.5%**		**62.1%**	**46.6%**		**33.3%**	**42.1%**		**64.7%**	**39.4%**	

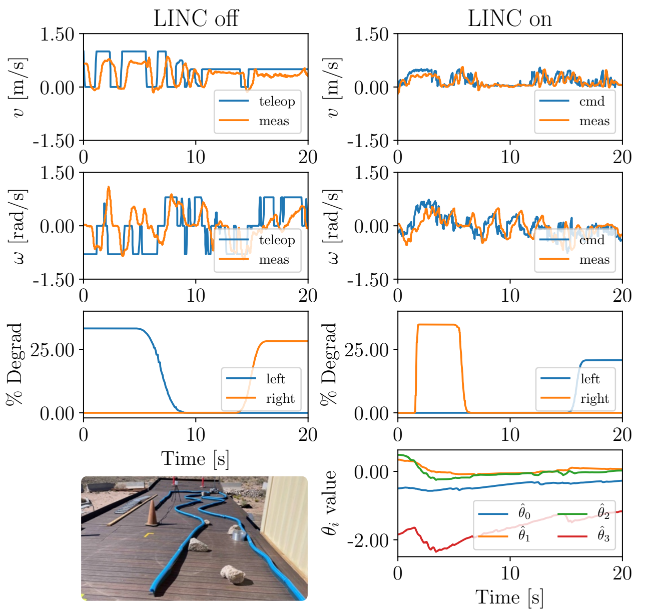


Fig. 20: Trajectory tracking during the Chicane Track. The right and left columns display tracking performance with and without our MAGIC controller, respectively.

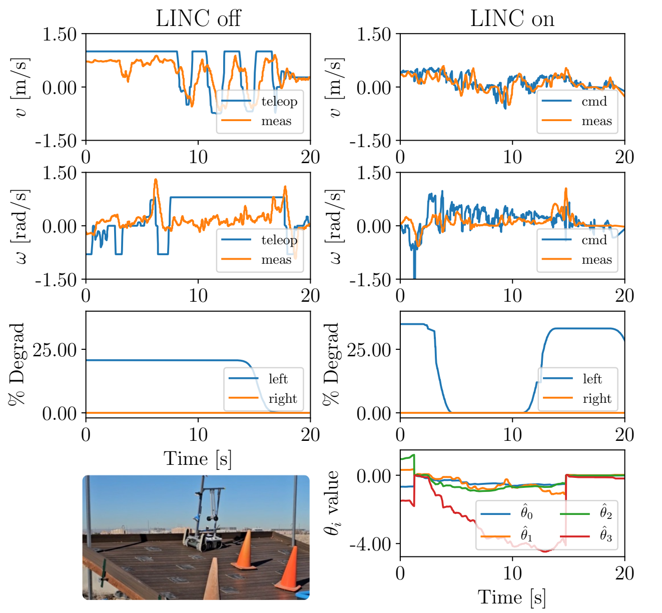


Fig. 21: Trajectory tracking during the Carpet Ramp. The right and left columns display tracking performance with and without our MAGIC controller, respectively.

collisions with the chicane walls. When MAGIC was activated, the robot navigated the chicane track more cautiously, moving approximately 2.5 times slower than with the baseline controller. This reduction in pace was a result of the software stack prioritizing the robot's safety by preventing collisions with walls.

2) Carpet Ramp: The goal of the Carpet Ramp exercise is to restore and maintain control under *track degradation* and *variable slippage*, all whilst mitigating the risk of tipping over. The ramp was composed of a slippery wooden surface with several patches of carpet to alter the ground friction coefficient, causing the tracks to slip asymmetrically. Additionally, as the roll angle of the robot increases over the incline, the traction of one of its two tracks is reduced as more of the weight falls over one of the tracks due to the high vertical center of gravity. This imbalance in traction causes the dynamics of the GVR-Bot to change significantly, especially affecting the ability to turn. This restricted turning behavior is shown in Fig. 21. The plot in the first column, second row shows that although the operator attempts to turn the GVR-Bot, very little control authority in angular velocity is achieved. By comparison, when operated with our MAGIC controller, the robot adapts to the terrain, tracking safer turn commands that reduce the risk of tipping (second column, second row). As seen in the bottom row of Fig. 21, the adaptation coefficients, especially the one for the angular velocity, greatly increase to compensate for slip. This particular exercise demonstrates the greatest improvement in performance relative to the baseline, exhibiting a 62.11% decrease in linear velocity error and a 46.55% decrease in angular velocity error, as seen in Table VII. This performance is attributed to the exercise's incorporation of both degradation and slippage elements.

3) Narrowing Corridor: The aim of the Narrowing Corridor mirrored that of the Chicane Track, emphasizing the robot's consistent navigation through a tight corridor despite track degradation. Its tracking performance can be seen in Fig. 23 and Table VII, in which we notice a significant decrease in tracking error for both angular and linear velocity.

4) Wedge Track: The Wedge Track tests the ability of the algorithms to maintain control and slow down under slippage while minimizing the risk of tipping over. When traversing discrete wooden wedges, the robot often loses traction. Moreover, the robot experiences sudden positive and negative accelerations in response to the downhill and uphill traversal of a wedge pair. The combination of these difficulties creates a challenging terrain to track velocities despite no degradation.

As shown in Fig. 22, with our controller's assistance and safe slowdowns from the planning, the robot can track velocities more accurately than without our MAGIC controller. By comparison, without assistance, the robot experiences large

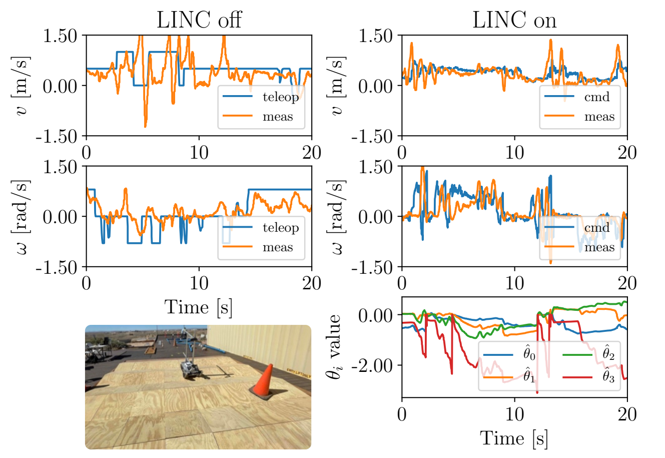


Fig. 22: Trajectory tracking during the Wedge Track run. The right and left columns display tracking performance with and without our controller, respectively. No artificial degradation was introduced by the organizers while the robot traversed wedges.

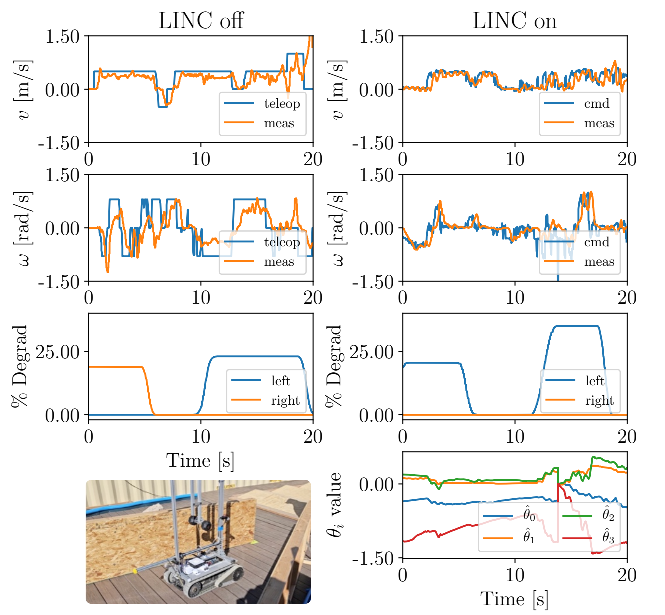


Fig. 23: Trajectory tracking during the narrow corridor run. The right and left columns display tracking performance with and without our controller, respectively.

velocity spikes as it traverses the wedges. These rapid changes are caused by the VIO's Kalman filter integrating spikes measured by the accelerometer when the robot bounces off the wedges. In the bottom row of Fig. 22, we also notice the adaptation coefficients quickly adapting for the loss of traction, especially for the angular velocity, and in Table VII, the decrease in tracking error, especially for the angular velocity tracking.

5) Driver Experience: The LINC program was different from many robotics projects in that the global planner (and to an extent, the local planner) was human-driven, rather than autonomous. This presents a special challenge as MAGIC must not degrade the user driving experience, and instead must augment the human driver without the forward-planning and control input smoothness assumptions of typical robotic projects that use a very large amount of autonomy. The following paragraph is written based on qualitative driver feedback from both an 'expert' (many hours of experience) and a 'novice' (no previous training or experience) driver.

The success of MAGIC in augmenting a human driver was twofold - firstly, MAGIC consistently ran fast enough such that there was no perceptible increase between joystick input and robot output, and secondly, much of the adaptation to the changing terrain and plant were significantly reduced by MAGIC. While the effects of the track degradation were noticeable, especially when abruptly applied, the apparent degradation was reduced by about 75% of the applied degradation within a second. This made driving significantly easier, especially during the tight chicane track sections with temporally varying degradation. For sections of the combined circuit where track slippage was present, MAGIC smoothed out much of the unwanted motion due to traction mismatch, yielding a far more predictable trajectory when driving. There were no noticeable delays introduced by MAGIC, nor were there any 'oscillations' in the traversed trajectories that adversely affected the driver's ability to operate the robot. MAGIC improved the driver experience both when joystick commands were directly fed to MAGIC, and when local trajectories were generated from joystick inputs by a local planner (MCTS). In both cases, during nominal operation (no track degradation, no track slippage) there was little-to-no noticeable change in vehicle behavior when dynamically switching in and out of the controller, indicating MAGIC accurately predicts and executes the desired behaviors generated from the joystick inputs.

VIII. Conclusion

We have introduced a novel learning-based composite adaptive controller that incorporates visual foundation models for terrain understanding and adaptation. The basis function of this adaptive controller, which is both state and terrain dependent, is learned offline using our proposed meta-learning algorithm. We prove the exponential convergence to a bounded tracking error ball of our adaptive controller and demonstrate that incorporating a pre-trained VFM into our learned representation enhances our controller's tracking performance compared to an equivalent controller without the learned representation. Our method showed a 53% improvement in position tracking error when deployed on a tracked vehicle traversing two different sloped terrains.

To gain insight into the inner workings of our full method, we empirically analyzed the features of the pre-trained VFM in terms of separability and continuity using support vector classifiers. This analysis showed positive empirical evidence that the Dino VFM is suitable for fine-grained discrimination of terrain types in images containing only terrain, and thus suitable for our control method.

We further tested our method under other perturbations, such as artificially induced track degradation. We demonstrated the effectiveness of our algorithm without terrain-aware basis function in human-in-the-loop driving scenarios. Our controller improved tracking of real-time human generated trajectories both in nominal and degraded vehicle states without introducing noticeable system delay. These experiments were part of the DARPA's Learning Introspective Control project.

Acknowledgement

We thank T. Touma for developing the hardware and the testing environments replicas; L. Gan and P. Proença and for the state estimation and elevation mapping; E. Sjögren for her early work on using Principal Component Analysis for analyzing the Dino features; Sandia National Laboratories team (T. Blada, D. Wood, E. Lu) for organizing the LINC tracks; J. Burdick for the LINC project management; A. Rahmani for the JPL Mars Yard support, LINC project management, and technical advice, and; Y. Yue, B. Riviere, and P. Spieler for stimulating discussions.

References

[1] M. M. Foglia and G. Reina, "Agricultural robot for radicchio harvesting," *Journal of Field Robotics*, vol. 23, pp. 363–377, July 2006.

[2] P. Gonzalez-De-Santos, R. Fernández, D. Sepúlveda, E. Navas, and M. Armada, *Unmanned Ground Vehicles for Smart Farms*, ch. 6, pp. 73–95. Intechopen, 2020.

[3] A. Bechar and C. Vigneault, "Agricultural robots for field operations: Concepts and components," *Biosystems Engineering*, vol. 149, pp. 94–111, June 2016.

[4] J. Delmerico, E. Mueggler, J. Nitsch, and D. Scaramuzza, "Active autonomous aerial exploration for ground robot path planning," *IEEE Robotics and Automation Letters*, vol. 2, pp. 664–671, Apr. 2017.

[5] Z. Kashino, G. Nejat, and B. Benhabib, "Aerial wilderness search and rescue with ground support," *Journal of Intelligent and Robotic Systems*, vol. 99, pp. 147–163, Jul. 2020.

[6] H. Qin *et al.*, "Autonomous exploration and mapping system using heterogeneous UAVs and UGVs in GPS-denied environments," *IEEE Trans. on Vehicular Technology*, vol. 68, pp. 1339–1350, Feb. 2019.

[7] A. Gadekar *et al.*, "Rakshak: A modular unmanned ground vehicle for surveillance and logistics operations," *Cognitive Robotics*, vol. 3, pp. 23–33, Mar. 2023.

[8] V. Verma *et al.*, "Autonomous robotics is driving Perseverance rover's progress on Mars," *Science Robotics*, vol. 8, Jul. 2023.

[9] R. E. Arvidson *et al.*, "Opportunity Mars Rover mission: Overview and selected results from Purgatory ripple to traverses to Endeavour crater," *Journal of Geophysical Research: Planets*, vol. 116, Feb. 2011.

[10] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars Exploration Rovers," *Journal of Field Robotics*, vol. 24, pp. 169–186, Mar. 2007.

[11] J. Y. Wong, *Theory of ground vehicles*. John Wiley & Sons, 2001.

[12] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. 32nd Int. Conf. on Neural Information Processing Systems*, pp. 4759–4770, Dec. 2018.

[13] M. Peng, B. Zhu, and J. Jiao, "Linear representation meta-reinforcement learning for instant adaptation," *arXiv preprint arXiv:2101.04750*, 2021.

[14] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 23–30, Sept. 2017.

[15] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proc. 33rd Int. Conf. on Machine Learning*, vol. 48, pp. 1842–1850, June 2016.

[16] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey, "Meta-learning in neural networks: A survey," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 5149–5169, Sept. 2021.

[17] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. on Machine Learning*, vol. 70, pp. 1126–1135, Aug. 2017.

[18] A. Nagabandi *et al.*, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *Int. Conf. on Learning Representations (ICLR)*, May 2019.

[19] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," in *Conf. on Robot Learning*, pp. 617–629, PMLR, 2018.

[20] I. Goodfellow *et al.*, "Generative Adversarial Networks," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[21] Y. Ganin *et al.*, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, pp. 2096–2030, May 2016.

[22] C. D. McKinnon and A. P. Schoellig, "Meta learning with paired forward and inverse models for efficient receding horizon control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3240–3247, 2021.

[23] J.-J. E. Slotine and W. Li, *Applied nonlinear control*. Englewood Cliffs, N.J: Prentice Hall, 1991.

[24] P. A. Ioannou and J. Sun, *Robust adaptive control*, vol. 1. Prentice-Hall Upper Saddle River, NJ, 1996.

[25] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos, *Nonlinear and adaptive control design*. John Wiley & Sons, Inc., 1995.

[26] K. S. Narendra and A. M. Annaswamy, *Stable adaptive systems*. Courier Corporation, 2012.

[27] M. O'Connell *et al.*, "Neural-Fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, May 2022.

[28] F.-C. Chen and H. K. Khalil, "Adaptive control of a class of nonlinear discrete-time systems using neural networks," *IEEE Trans. on Automatic Control*, vol. 40, no. 5, pp. 791–801, 1995.

[29] J. A. Farrell and M. M. Polycarpou, *Adaptive Approximation Based Control*. John Wiley & Sons, Ltd, 2006.

[30] J. Nakanishi, J. Farrell, and S. Schaal, "A locally weighted learning composite adaptive controller with structure adaptation," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 1, pp. 882–889 vol.1, Sept. 2002.

[31] J. Pravitra, K. A. Ackerman, C. Cao, N. Hovakimyan, and E. A. Theodorou, "L1-Adaptive MPPI Architecture for Robust and Agile Control of Multirotors," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 7661–7666, Oct. 2020.

[32] X. Song *et al.*, "Rapidly adaptable legged robots via evolutionary meta-learning," 2020.

[33] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine, "Model-based meta-reinforcement learning for flight with suspended payloads," *IEEE Robotics and Automation Letters*, vol. 6, pp. 1471–1478, Apr. 2021.

[34] C. D. McKinnon and A. P. Schoellig, "Meta learning with paired forward and inverse models for efficient receding horizon control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3240–3247, 2021.

[35] B. Rothrock, R. Kennedy, C. Cunningham, J. Papon, M. Heverly, and M. Ono, "SPOC: Deep learning-based terrain classification for Mars Rover missions," American Institute of Aeronautics and Astronautics (AIAA) Forum, Sept. 2016.

[36] J. Schmidhuber, "Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook," Diploma Thesis, Technische Universität Munchen, Germany, May 1987.

[37] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," 2017.

[38] G. Shi, K. Azizzadenesheli, M. O'Connell, S.-J. Chung, and Y. Yue, "Meta-adaptive nonlinear control: Theory and algorithms," 2021.

[39] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," 2020.

[40] W. Xiao, T. He, J. Dolan, and G. Shi, "Safe deep policy adaptation," *arXiv preprint arXiv:2310.08602*, 2023.

[41] C. Finn, A. Rajeswaran, S. M. Kakade, and S. Levine, "Online meta-learning," *CoRR*, vol. abs/1902.08438, 2019.

[42] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," *CoRR*, vol. abs/1710.03641, 2017.

[43] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," in *Proc. 36th Int. Conf. on Machine Learning*, vol. 97, pp. 1920–1930, June 2019.

[44] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *CoRR*, vol. abs/1703.03400, 2017.

[45] J. Harrison, A. Sharma, C. Finn, and M. Pavone, "Continuous meta-learning without tasks," *CoRR*, vol. abs/1912.08866, 2019.

[46] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, 1992.

[47] D. Ha and J. Schmidhuber, "World models," *CoRR*, vol. abs/1803.10122, 2018.

[48] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[49] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "Daydreamer: World models for physical robot learning," in *Proc. 6th Conf. on Robot Learning*, vol. 205, pp. 2226–2240, Dec. 2023.

[50] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," 2023.

[51] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 5026–5033, Oct. 2012.

[52] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.

[53] L. Gan, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, "Energy-based legged robots terrain traversability modeling via deep inverse reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, p. 8807–8814, Oct. 2022.

[54] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, "Fast traversability estimation for wild visual navigation," IEEE International Conference on Robotics and Automation Workshop on Pretraining4Robotics, 2023.

[55] M. Oquab *et al.*, "DINOv2: Learning robust visual features without supervision," 2023.

[56] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[57] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020.

[58] M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, and W. T. Freeman, "Unsupervised semantic segmentation by distilling feature correspondences," in *Int. Conf. on Learning Representations*, 2022.

[59] G. Erni, J. Frey, T. Miki, M. Mattamala, and M. Hutter, "MEM: Multi-Modal elevation mapping for robotics and learning," 2023.

[60] M. Naseer, K. Ranasinghe, S. Khan, M. Hayat, F. Khan, and M.-H. Yang, "Intriguing properties of vision transformers," in *Advances in Neural Information Processing Systems*, 2021.

[61] A. M. Annaswamy and A. L. Fradkov, "A historical perspective of adaptive control and learning," *Annu. Reviews in Control*, vol. 52, pp. 18–41, Oct. 2021.

[62] X. Shi, P. Spieler, E. Tang, E.-S. Lupu, P. Tokumaru, and S.-J. Chung, "Adaptive Nonlinear Control of Fixed-Wing VTOL with Airflow Vector Sensing," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 5321–5327, May 2020.

[63] V. Kadirkamanathan and S. Fabri, "Stable nonlinear adaptive control with growing radial basis function networks," *5th IFAC Symp. on Adaptive Systems in Control and Signal Processing*, vol. 28, pp. 245–250, June 1995.

[64] C. A. Lúa, D. Bianchi, and S. Di Gennaro, "Nonlinear observer-based adaptive control of ground vehicles with uncertainty estimation," *Journal of the Franklin Institute*, vol. 360, no. 18, pp. 14175–14189, 2023.

[65] A. Mohammadzadeh and H. Taghavifar, "A novel adaptive control approach for path tracking control of autonomous vehicles subject to uncertain dynamics," *Proc. Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 234, no. 8, pp. 2115–2126, 2020.

[66] M. Sombolestan, Y. Chen, and Q. Nguyen, "Adaptive force-based control for legged robots," 2021.

[67] K. Tsujita, K. Tsuchiya, and A. Onat, "Adaptive gait pattern control of a quadruped locomotion robot," in *Proc. 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 4, pp. 2318–2325 vol.4, 2001.

[68] P. Petrov and F. Nashashibi, "Modeling and nonlinear adaptive control for autonomous vehicle overtaking," *IEEE Trans. on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1643–1656, 2014.

[69] S. Khan and J. Guivant, "Fast nonlinear model predictive planner and control for an unmanned ground vehicle in the presence of disturbances and dynamic obstacles," *Scientific Reports*, vol. 12, p. 12135, July 2022.

[70] M. Visca, R. Powell, Y. Gao, and S. Fallah, "Deep meta-learning energy-aware path planner for unmanned ground vehicles in unknown terrains," *IEEE Access*, vol. 10, pp. 30055–30068, 2022.

[71] J. Seo, T. Kim, S. Ahn, and K. Kwak, "Metaverse: Meta-learning traversability cost map for off-road navigation," 07 2023.

[72] X. Cai, M. Everett, L. Sharma, P. Osteen, and J. How, "Probabilistic traversability model for risk-aware motion planning in off-road environments," *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2023.

[73] M. O'Connell, *Methods for Robust Learning-Based Control*. PhD thesis, 2023.

[74] G. Shi *et al.*, "Neural lander: Stable drone landing control using learned dynamics," in *Int. Conf. on Robotics and Automation*, pp. 9784–9790, Mar. 2019.

[75] H. Tsukamoto, S.-J. Chung, and J.-J. E. Slotine, "Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview," *Annu. Reviews in Control*, vol. 52, pp. 135–169, Oct. 2021.

[76] J.-J. Slotine and W. Li, *Applied nonlinear control*. Prentice Hall, 1991.

[77] N. Strawa, D. I. Ignatyev, A. C. Zolotas, and A. Tsourdos, "On-line learning and updating unmanned tracked vehicle dynamics," *Electronics*, vol. 10, Jan. 2021.

[78] R. Byrne and C. Abdallah, "Design of a model reference adaptive controller for vehicle road following," *Mathematical and Computer Modelling*, vol. 22, no. 4, pp. 343–354, 1995.

[79] M. Mistry and L. Righetti, "Operational Space Control of Constrained and Underactuated Systems," in *Proc. of Robotics: Science and Systems*, June 2011.

[80] F. Aghili, "A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: applications to control and simulation," *IEEE Trans. on Robotics*, vol. 21, pp. 834–849, Oct. 2005.

[81] L. Caracciolo, A. De Luca, and S. Iannitti, "Trajectory tracking control of a four-wheel differentially driven mobile robot," *IEEE Int. Conf. on Robotics and Automation*, May 1999.

[82] H. Pacejka, *Tire and vehicle dynamics, 2nd Edition*. Elsevier, 2006.

[83] H. K. Khalil, *Nonlinear Systems, 3rd Edition*. Prentice Hall, 2002.

[84] S.-J. Chung, S. Bandyopadhyay, I. Chang, and F. Y. Hadaegh, "Phase synchronization control of complex networks of lagrangian systems on adaptive digraphs," *Automatica*, vol. 49, no. 5, pp. 1148–1161, 2013.

[85] D. Simon, *Optimal State Estimation*, ch. 11, pp. 331–371. John Wiley & Sons, Inc., 2006.

[86] M. Caron *et al.*, "Emerging properties in self-supervised vision transformers," in *Proc. Int. Conf. on Computer Vision*, 2021.

[87] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google Landmarks Dataset v2 - A large-scale benchmark for instance-level recognition and retrieval," *Computer Vision and Pattern Recognition*, Nov. 2020.

[88] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, pp. 18–28, July 1998.

[89] R. Volpe, "JPL Robotics: The MarsYard III," 2023. Accessed: 2024-01-11.

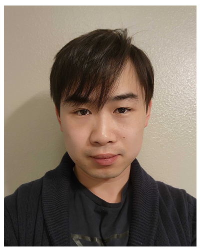
[90] US ARMY CCDC GVSC, "GVR-BOT Users Guide," 2019.

[91] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A research platform for visual-inertial estimation," *IEEE Int. Conf. on Robotics and Automation*, May 2020.

[92] DARPA, "Learning introspective control (LINC)." https://www.darpa.mil/program/learning-introspective-control.

[93] J. Lathrop, J. A. Preiss, E. S. Lupu, F. Xie, and S.-J. Chung, "Improved autonomy with rapidly learned dynamics from adaptive control." 2024.

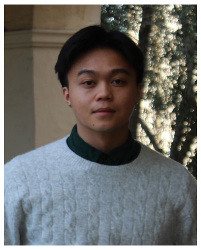
Elena Sorina Lupu is a PhD student in the Aerospace Department at California Institute of Technology and an affiliate of the Keck Institute of Space Sciences. She has master degrees from Caltech in Space Engineering and from the Swiss Federal Institute of Technology, Lausanne (EPFL) in Robotics and Autonomous Systems. Her current research focuses on autonomy, control, and machine learning applied to robotics and spacecraft. She won awards and fellowships like the AI4Science Fellowship from Amazon, the Anita Borg Women Techmakers from Google, and the Best Student Paper Award at the 11^{th} International Workshop on Satellite Constellations and Formation Flying.



Fengze Xie is a PhD student in the Department of Computing and Mathematical Sciences at the California Institute of Technology. He received a M.S. in Electrical Engineering from the California Institute of Technology in 2021, following the completion of a B.S. in Computer Engineering and a B.S. in Engineering Physics from the University of Illinois Urbana-Champaign in 2020. His research interests mainly focus on the intersection between machine learning and robotics, including control, planning, and perception. He was the recipient of the Simoudis Discovery Prize.

James Alan Preiss received the Ph.D. degree in computer science from the University of Southern California in 2022 and the B.A./B.S. degree with concentrations in mathematics and photography from The Evergreen State College in 2010. He is currently a Postdoctoral Scholar Research Associate in the Department of Computing + Mathematical Sciences at the California Institute of Technology, Pasadena, CA, USA. He has held positions as a Research Intern at Google Research, Software Engineer at SAS Institute, Associate Software Engineer at Geomagic/3D Systems, and Research Technician at Barlow Scientific. His research interests focus on developing rigorous foundations for the application of machine learning to planning and control in robotics.

Matthew Anderson received a B.Eng (Aeronautical, Hons I) from The University of Sydney in 2010 and a jointly awarded Ph.D from The University of Sydney and Université libre de Bruxelles in 2018. He is currently a Staff Scientist in the Computing + Mathematical Sciences (CMS) department at the California Institute of Technology and supports a wide variety of robotics research with a fleet of UAVs and UGVs. His primary areas of research focus on UAV operations, aircraft modelling, and extreme environment robotics.



Jedidiah Alindogan is a research engineering staff at Caltech. He received his B.S. in mechanical engineering from Caltech in 2023. At Caltech, he along with a team of undergraduates submitted to the NASA 2022 BIG Idea challenge and was awarded the Best Visionary Concept Award. Finally, he was granted membership to the Tau Beta Phi engineering honor society for academic achievement as well as a commitment to personal and professional integrity.

Soon-Jo Chung (M'06–SM'12) received the S.M. degree in aeronautics and astronautics and the Sc.D. degree in estimation and control from Massachusetts Institute of Technology, Cambridge, MA, USA, in 2002 and 2007, respectively. He is currently Bren Professor of Control and Dynamical Systems, and a Jet Propulsion Laboratory Senior Research Scientist at the California Institute of Technology, Pasadena, CA, USA. He was an Associate Editor of IEEE TRANSACTIONS ON AUTOMATIC CONTROL and IEEE TRANSACTIONS ON ROBOTICS, and the Guest Editor of a Special Section on Aerial Swarm Robotics published in IEEE TRANSACTIONS ON ROBOTICS.