MAGIC^{VFM} -Meta-learning Adaptation for Ground Interaction Control with Visual Foundation Models

Author Names Omitted for Anonymous Review.

***Abstract*—Control of off-road vehicles is challenging due to the complex dynamic interactions with the terrain. Accurate modeling of these interactions is important to optimize driving performance, but the relevant physical phenomena, such as slip, are too complex to model from first principles. Therefore, we present an offline meta-learning algorithm to construct a rapidly-tunable model of residual dynamics and disturbances. Our model processes terrain images into features using a visual foundation model (VFM), then maps these features and the vehicle state to an estimate of the current actuation matrix using a deep neural network (DNN). We then combine this model with composite adaptive control to modify the last layer of the DNN in real time, accounting for the remaining terrain interactions not captured during offline training. We provide mathematical guarantees of stability and robustness for our controller, and demonstrate the effectiveness of our method through simulations and hardware experiments with a tracked vehicle and a car-like robot. We evaluate our method outdoors on different slopes with varying slippage and actuator degradation disturbances, and compare against an adaptive controller that does not use the VFM terrain features. We show significant improvement over the baseline in both hardware experimentation and simulation.**

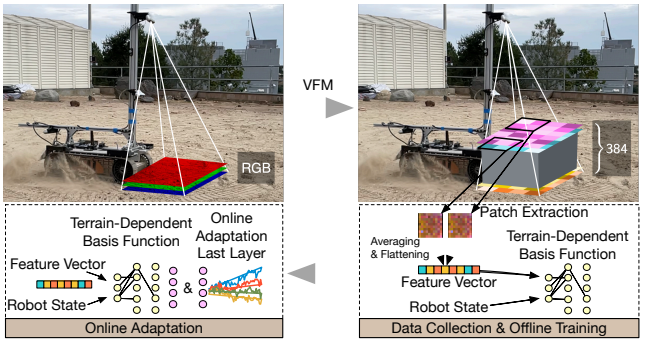


Fig. 1: MAGIC^{VFM}: An offline meta-learning algorithm to build a residual dynamics and disturbance model using both Visual Foundation Models (VFM) and vehicle states. This model is integrated with composite adaptive control to adapt to changes in both the terrain and vehicle dynamics conditions in real time.

I. INTRODUCTION

To better understand the effects of terradynamics, researchers have designed sophisticated models [Wong(2001)] 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. In addition, designing controllers that consider these complex models is challenging. For control, kinematic models, such as Dubins, 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.

Furthermore, 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 [Rothrock et al.(2016)]. 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 yet induce different dynamic behaviors on the robot (e.g., deep and shallow sand). Therefore, finding the most accurate and robust representation of the environment is indispensable for vehicle control over complex terrain.

A. Contributions

We present MAGIC^{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 VFM with meta-learning and composite adaptive control, thereby enabling ground vehicles to navigate and adapt to complex terrains in real time. Our method is well-suited for any ground vehicle equipped with the following: 1) sensors to measure the internal robot 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 in real time;
- 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 and robustness against model errors for adaptive control with visual

and state input that works in conjunction with our 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 and arbitrary time-varying disturbances.

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

II. RELATED WORK

A. Meta-learning

The term *meta-learning*, first coined in [Schmidhuber(1987)], most often refers to learning protocols in which there is an underlying set of related learning tasks/environments, and the learner leverages data from previously seen tasks to adapt rapidly to a new task [Finn et al.(2017)Finn, Abbeel, and Levine], [Shi et al.(2021)Shi, Azizzadenesheli, O'Connell, Chung, and Yue], [Hospedales et al.(2021)Hospedales, Antoniou, Micaelli, and Storkey], [Xiao et al.(2024)Xiao, He, Dolan, and Shi]. 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 [Nagabandi et al.(2019)], [Song et al.(2020)], [Belkhale et al.(2021)], [McKinnon and Schoellig(2021)]. Some examples of meta-learning algorithms from the literature are Model-Agnostic Meta-Learning (MAML) [Finn et al.(2017)Finn, Abbeel, and Levine] with its online extension [Finn et al.(2019)Finn, Rajeswaran, Kakade, and Levine], Meta-learning via Online Changepoint Analysis (MOCA) [Harrison et al.(2020)Harrison, Sharma, Finn, and Pavone], and Domain Adversarially Invariant Meta-Learning (DAIML) used in Neural-Fly [O'Connell et al.(2022)].

Our method builds on the previous work [O'Connell et al.(2022)] on the integration of adaptive control and offline meta-learning to create 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.

B. Embedding Visual Information in Classical Control and Reinforcement Learning

Recently, vision-based reinforcement learning (VRL) has demonstrated the capability to control agents in simulated environments [Ha and Schmidhuber(2018)], as well as robots in real environments, with applications to ground robots [Miki et al.(2022)Miki, Lee, Hwangbo, Wellhausen, Koltun, and Hutter], [Wu et al.(2023)Wu, Escontrela, Hafner, Abbeel, and Goldberg], [Cheng et al.(2024)Cheng, Shi, Agarwal, and Pathak] and robot manipulation [Levine et al.(2016)Levine, Finn, Darrell, and Abbeel], [Florence et al.(2019)Florence, Manuelli, and Tedrake], [Ebert et al.(2018)Ebert, Finn, Dasari, Xie, Lee, and Levine]. This capability is achieved by leveraging high-fidelity models in robotics simulators [Todorov et al.(2012)Todorov, Erez, and Tassa], [Hwangbo et al.(2018)Hwangbo, Lee, and Hutter], imitation learning from human demonstrations or techniques to bridge the sim-to-real gap of the learned policy [Tobin et al.(2017)Tobin, Fong, Ray, Schneider, Zaremba, and Abbeel]. Nevertheless, a limitation of VRL is that the generated policy remains uninterpretable and does not have safety and robustness guarantees. Despite progress in combining vision with reinforcement learning, 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 address this limitation, we derive a nonlinear adaptive tracking controller with stability guarantees for AGVs that uses a learned ground model with vision information incorporated in the control influence matrix.

See Sec. II in Appendix I for a more comprehensive literature review.

III. METHODS

A. Residual Dynamics Representation using VFMs

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

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

where $\boldsymbol{\theta}(t) = [\theta_1 \dots \theta_{n_\theta}]^\top \in \mathbb{R}^{n_\theta}$ denotes a time-varying vector of linear parameters that are adapted online by our algorithm, $\boldsymbol{\delta} \in \mathbb{R}^n$ is a representation error, 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 the adaptive control part of our method, the precise form of $\boldsymbol{\mathcal{E}}$ is not required in our work. We can think of $\boldsymbol{\mathcal{E}}$ as computed by some arbitrary Lipschitz function from the robot's sensors to $\mathbb{R}^p$. The feature mapping $\boldsymbol{\Phi}(\mathbf{x}, \mathbf{u}, \boldsymbol{\mathcal{E}}) : \mathbb{R}^{n+m+p} \mapsto \mathbb{R}^{n \times n_\theta}$ 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 $\boldsymbol{\Phi}$ is a DNN. In this case, $\boldsymbol{\theta}$ can be regarded as the weights of the last layer of the DNN (2), which continuously adapt in real-time. Further, we assume that the disturbance $\mathbf{d}$ is affine in the control input $\mathbf{u}$, taking the form

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

where the dependence on a parameter vector $\mathbf{w}$ (the DNN weights) is made explicit, and the basis function has the form

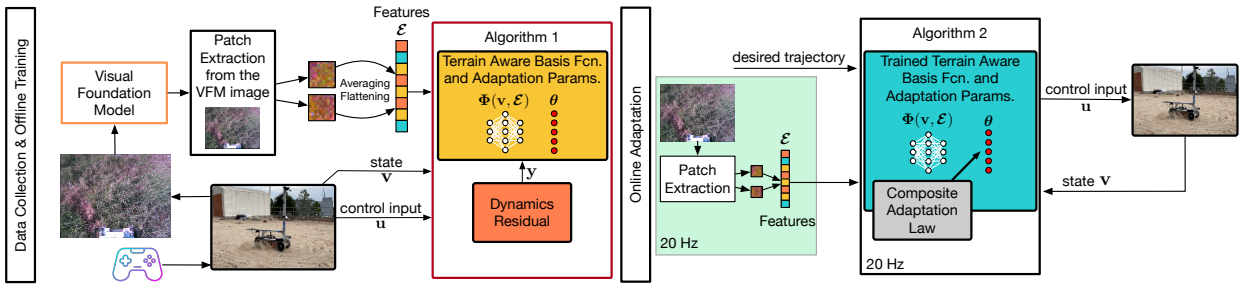


Fig. 2: Terrain-aware Architecture: offline data collection and training, followed by real-time adaptive control running onboard the robot.

$\mathbf{\Phi}^{\mathbf{w}}(\mathbf{x}, \boldsymbol{\mathcal{E}}) : \mathbb{R}^{n+p} \mapsto \mathbb{R}^{n \times m \times n_\theta}$ with the individual matrix-valued components denoted by $\mathbf{\Phi}_i^{\mathbf{w}}(\mathbf{x}, \boldsymbol{\mathcal{E}}) : \mathbb{R}^{n+p} \mapsto \mathbb{R}^{n \times m}$. The control affine assumption is motivated by two factors presented in Sec. III-A of Appendix I. Lastly, we define the dynamics residual that is used in both the online and offline phase of our method as $\mathbf{y} = \dot{\mathbf{x}} - \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$.

B. Offline Meta-Learning Phase

In Fig.2, we illustrate the structure of our proposed solution to learn offline the basis function $\mathbf{\Phi}^{\mathbf{w}}(\mathbf{x}, \boldsymbol{\mathcal{E}})$ in (3) and to make real-time adjustments using composite adaptive control (Sec. III-C in the Appendix I). 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 $\mathbf{\Phi}^{\mathbf{w}}$ in (3), we collect a dataset of the robot operating 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 representation $\boldsymbol{\mathcal{E}}$, as discussed in Sec. III-A. This dataset contains $N \in \mathbb{N}$ trajectories. Each trajectory is an uninterrupted driving session on the order of a few minutes. The details of the Dataset and the Model Architecture is in Sec. III-B of the Appendix.

2) Optimization: Our method is built around the assumption that two terrains with similar visual features $\boldsymbol{\mathcal{E}}$ will usually, but not always, induce similar dynamics. We account for this observation 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 $\boldsymbol{\theta}$ 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. 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

$$J(\mathbf{w}) = \mathop{\mathbb{E}}_{n,\hat{\ell},s} \left[\sum_{t=s}^{s+\hat{\ell}-1} \left\| \mathbf{y}_t^n - \left(\mathbf{\Phi}^{\mathbf{w}}(\mathbf{x}_t^n, \boldsymbol{\mathcal{E}}_t^n) \boldsymbol{\theta}^\star_{n,\hat{\ell},s}(\mathbf{w}) \right) \mathbf{u}_t^n \right\|_2^2 \right] := \mathop{\mathbb{E}}_{n,\hat{\ell},s} \left[J_{n,\hat{\ell},s}(\mathbf{w}) \right], \tag{4}$$

where the expectation is shorthand for $n \sim \mathcal{U}[1, N]$, $\hat{\ell} \sim L$, and $s \sim \mathcal{U}[1, \ell - \hat{\ell} + 1]$. We incorporate the closed-form computation of the best-fit linear component $\boldsymbol{\theta}^\star_{n,\hat{\ell},s}$ in the computational graph of our optimization , as opposed to treating the trajectories of $\boldsymbol{\theta}$ as optimization variables. Given this, we obtain a simpler algorithm. Our offline training procedure details is given in Algorithm 1 in Appendix 1. It consists of stochastic first-order optimization on the objective $J(\mathbf{w})$ and spectral normalization to enforce the Lipschitz constraint on $\mathbf{\Phi}$. For the remaining sections, the parameters $\mathbf{w}$ of the basis function $\mathbf{\Phi}$ are fixed. Therefore, we drop the superscript and refer to $\mathbf{\Phi}^{\mathbf{w}}$ as $\mathbf{\Phi}$, for simplicity of notation. For our empirical work, we selected DINO V1 [Caron et al.(2021)Caron, Touvron, Misra, J'egou, Mairal, Bojanowski, and Joulin] 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. In Sec. V in Appendix 1, we provide an analysis of the features of the VFM. 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 control setting.

IV. System Modelling and Control Synthesis

We apply the methods from Algorithms 1 and 2 in Appendix 1 to a skid-steering tracked vehicle (Fig. 4 in Appendix 1) (Sec. IV-A through IV-C in Appendix 1) and to an Ackermann-steering vehicle (Sec. IV-D through IV-E in Appendix 1). The 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 DOF dynamics model (see (10) in Appendix 1) with its corresponding simplified model of the form (see (11) in Appendix 1). To this simplified model, we

apply an adaptive controller with learned ground information of the form (18)-(19) in Appendix 1. The car-like vehicle uses the Ackermann steering geometry. For this vehicle type, we derive a 3-DOF dynamics model (28) in Appendix 1 using the bicycle model and an adaptive controller using Algorithm 2 in Appendix 1. See Sec. IV in Appendix 1 for a detailed description. Lastly, we prove that tracking errors and the parameter errors exponentially converge to a bounded error ball. The details of the proof can be found in Sec. IV in Appendix 1.

V. Empirical Results: Hardware Experiments

We focus on the hardware implementation and experimental validation of our MAGIC$^{\text{VFM}}$ adaptive controller on a tracked vehicle, called GVR-Bot, and a car with Ackermann steering. We present how our adaptive controller effectively addresses various perturbations such as terrain changes, severe track degradation, and unknown internal robot dynamics.

A. Experiments on Slopes

To verify the performance of our MAGIC$^{\text{VFM}}$ controller on different terrains, the GVR-Bot tracked vehicle was driven on different slopes. Fig.3 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 in Appendix 1. By leveraging the strengths of a pre-trained VFM, we develop the lightweight DNN basis function head used in the adaptive controller of (18) and (19) from Appendix 1.

2) Online Adaptation: The benefits of the terrain-informed basis function can been seen by comparing the performance of a constant and non-constant basis function controller as the robot traverses slopes. Each experiment was carried out five times, with the results detailed in Fig. 13 of Appendix 1. The desired trajectory is a straight line that spans the entire length of the two slopes. We show that when the robot traverses the first slope (flagstone resulting in minimal slippage), both controllers have comparable tracking errors. However, a significant 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 of Appendix 1, we quantify the performance. The results demonstrate that the integration of a VFM in an adaptive control framework enhances tracking performance, yielding an average improvement of 53%. Computational load is a significant bottleneck when deploying VFMs onboard robots. Therefore, to minimize inference time and allow high controller rates, we employ the smallest transformer architecture of the DINO V1, which enables the controller to run at 20 Hz on the Graphics Processing Unit (GPU) on-board an NVIDIA Jetson Orin.

B. Experiments On-board an Ackermann Steering Vehicle

We performed similar experiments to those described in Sec. V-A using an Ackermann steering vehicle. Here, the robot traverses two different terrains, as seen in Fig.3, which induce different dynamic behaviors onto the robot (grass is more slippery than concrete). In Fig. 14 in Appendix 1, we show the product $\mathbf{\Phi}\hat{\boldsymbol{\theta}}$ for the constant basis function of the nonlinear tracking controller in (36) of Appendix 1. As the robot transitions between the two terrains, we see that the robot effectively adapts to each terrain during this transition. This behavior mirrors that observed in the simulation plots (Fig. 8 in Appendix 1). as the robot transitions between the two terrains (grass and concrete) several times. The DNN basis function switches depending on the type of environment it operates in, while the corresponding adaptation coefficients $\hat{\boldsymbol{\theta}}$ remain mostly constant. This behavior also mirrors that observed in the simulation plots (Fig. 8 of Appendix 1) when a DNN with a VFM is employed. Lastly, in Sec. VII-E in Appendix 1, we show the performance of our algorithm running a constant basis function on-board the GVR-Bot tracked vehicle during the DARPA's Learning Introspective Control. Details of the hardware experiments can be found in Sec. VII-E in Appendix 1.

VI. Conclusion

We introduced a 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% decrease in position tracking error when deployed on a tracked vehicle traversing two different sloped terrains. We further demonstrated our algorithm onboard a car-like vehicle and showed that the learned DNN basis function captures the residual dynamics generated by the two different terrains.

Fig. 3: a) The GVR-Bot traversing two slopes with different textures and terrain-induced dynamic behaviors. b) 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. c) The Traxxas robot traversing two different terrains that induce different dynamic behaviors. d) The Traxxas robot with its main sensing and compute units highlighted.

REFERENCES

[Belkhale et al.(2021)] Suneel Belkhale et al. Model-based meta-reinforcement learning for flight with suspended payloads. *IEEE Robot. Autom. Letters*, 6(2):1471–1478, April 2021. doi: 10.1109/lra.2021.3057046. URL https://doi.org/10.1109%2Flra.2021.3057046.

[Caron et al.(2021)Caron, Touvron, Misra, J'egou, Mairal, Bojanowski, and Joulin] Mathilde Caron, Hugo Touvron, Ishan Misra, Herv'e J'egou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021. URL https://api.semanticscholar.org/CorpusID:233444273.

[Cheng et al.(2024)Cheng, Shi, Agarwal, and Pathak] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450, 2024. doi: 10.1109/ICRA57147.2024.10610200.

[Ebert et al.(2018)Ebert, Finn, Dasari, Xie, Lee, and Levine] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex X. Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *CoRR*, abs/1812.00568, 2018. URL http://arxiv.org/abs/1812.00568.

[Finn et al.(2017)Finn, Abbeel, and Levine] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. 34th Int. Conf. Machine Learning*, volume 70, pages 1126–1135, August 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

[Finn et al.(2019)Finn, Rajeswaran, Kakade, and Levine] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *Proc. 36th Int. Conf. on Machine Learning*, volume 97, pages 1920–1930, June 2019.

[Florence et al.(2019)Florence, Manuelli, and Tedrake] Peter R. Florence, Lucas Manuelli, and Russ Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robot. Autom. Letters*, 5:492–499, 2019. URL https://api.semanticscholar.org/CorpusID:202578098.

[Ha and Schmidhuber(2018)] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2451–2463. 2018.

[Harrison et al.(2020)Harrison, Sharma, Finn, and Pavone] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Syst.*, volume 33, pages 17571–17581. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/cc3f5463bc4d26bc38eadc8bcffbc654-Paper.pdf.

[Hospedales et al.(2021)Hospedales, Antoniou, Micaelli, and Storkey] Timothy M Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44:5149–5169, September 2021. doi: 10.1109/TPAMI.2021.3079209.

[Hwangbo et al.(2018)Hwangbo, Lee, and Hutter] Jemin Hwangbo, Joonho Lee, and Marco Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robot. Autom. Letters*, 3(2):895–902, 2018. doi: 10.1109/LRA.2018.2792536.

[Levine et al.(2016)Levine, Finn, Darrell, and Abbeel] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Machine Learning Research*, 17(39):1–40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

[McKinnon and Schoellig(2021)] Christopher D. McKinnon and Angela P. Schoellig. Meta learning with paired forward and inverse models for efficient receding horizon control. *IEEE Robot. Autom. Letters*, 6(2):3240–3247, 2021. doi: 10.1109/LRA.2021.3063957.

[Miki et al.(2022)Miki, Lee, Hwangbo, Wellhausen, Koltun, and Hutter] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62), 2022. doi: 10.1126/scirobotics.abk2822. URL https://www.science.org/doi/abs/10.1126/scirobotics.abk2822.

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

[O'Connell et al.(2022)] Michael O'Connell et al. Neural-Fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66), May 2022. doi: 10.1126/scirobotics.abm6597. URL https://doi.org/10.1126%2Fscirobotics.abm6597.

[Rothrock et al.(2016)] Brandon Rothrock et al. SPOC: Deep learning-based terrain classification for Mars Rover missions. In *AIAA SPACE 2016*. Amer. Inst. Aeronaut. Astronautics (AIAA) Forum, September 2016. ISBN 978-1-62410-427-5. doi: 10.2514/6.2016-5539. URL https://arc.aiaa.org/doi/10.2514/6.2016-5539.

[Schmidhuber(1987)] Jurgen 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. URL http://www.idsia.ch/~juergen/diploma.html.

[Shi et al.(2021)Shi, Azizzadenesheli, O'Connell, Chung, and Yue] Guanya Shi, Kamyar Azizzadenesheli, Michael O'Connell, Soon-Jo Chung, and Yisong Yue. Meta-adaptive nonlinear control: Theory and algorithms. *Advances Neural Information Processing Systems*, 2021.

[Song et al.(2020)] Xingyou Song et al. Rapidly adaptable legged robots via evolutionary meta-learning. *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, October 2020.

[Tobin et al.(2017)Tobin, Fong, Ray, Schneider, Zaremba, and Abbeel] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ Int. Conf. Intelligent Robots Systems*, pages 23–30, September 2017.

[Todorov et al.(2012)Todorov, Erez, and Tassa] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based

control. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 5026–5033, October 2012. doi: 10.1109/IROS.2012.6386109.

[Wong(2001)] Jo Yung Wong. *Theory of ground vehicles*. John Wiley & Sons, 2001.

[Wu et al.(2023)Wu, Escontrela, Hafner, Abbeel, and Goldberg] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Proc. 6th Conf. on Robot Learning*, volume 205, pages 2226–2240, December 2023. URL https://proceedings.mlr.press/v205/wu23c.html.

[Xiao et al.(2024)Xiao, He, Dolan, and Shi] Wenli Xiao, Tairan He, John Dolan, and Guanya Shi. Safe deep policy adaptation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 17286–17292, 2024. doi: 10.1109/ICRA57147.2024.10611340.

APPENDIX I

See next page.

MAGIC^{VFM} -Meta-learning Adaptation for Ground Interaction Control with Visual Foundation Models

***Abstract*—Control of off-road vehicles is challenging due to the complex dynamic interactions with the terrain. Accurate modeling of these interactions is important to optimize driving performance, but the relevant physical phenomena, such as slip, are too complex to model from first principles. Therefore, we present an offline meta-learning algorithm to construct a rapidly-tunable model of residual dynamics and disturbances. Our model processes terrain images into features using a visual foundation model (VFM), then maps these features and the vehicle state to an estimate of the current actuation matrix using a deep neural network (DNN). We then combine this model with composite adaptive control to modify the last layer of the DNN in real time, accounting for the remaining terrain interactions not captured during offline training. We provide mathematical guarantees of stability and robustness for our controller, and demonstrate the effectiveness of our method through simulations and hardware experiments with a tracked vehicle and a car-like robot. We evaluate our method outdoors on different slopes with varying slippage and actuator degradation disturbances, and compare against an adaptive controller that does not use the VFM terrain features. We show significant improvement over the baseline in both hardware experimentation and simulation.**

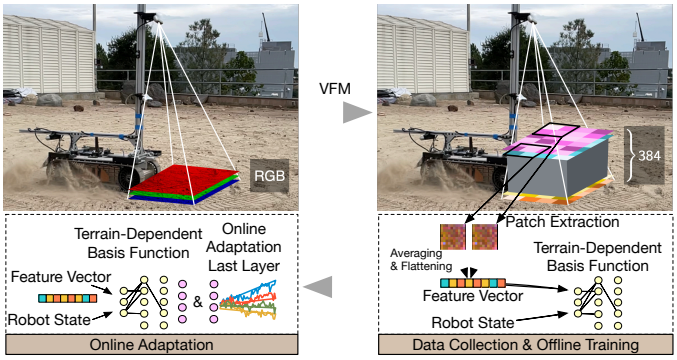


Fig. 1: MAGIC^{VFM}: An offline meta-learning algorithm to build a residual dynamics and disturbance model using both Visual Foundation Models (VFM) and vehicle states. This model is integrated with composite adaptive control to adapt to changes in both the terrain and vehicle dynamics conditions in real time. See https://youtu.be/sxM73ryweRA

I. Introduction

Autonomous Ground Vehicles (AGVs) are gaining popularity across numerous domains including agriculture applications [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 degrades. 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 encountered by ground vehicles while operating on loose terrain. 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 rerouted 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 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, 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.

Controllers that 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 by the controller. Recently, 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, efficiently addressing 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 yet induce different dynamic behaviors on the robot (e.g., deep and shallow sand). Therefore, finding the most accurate and robust representation of the environment is indispensable for vehicle control over complex terrain.

A. Contributions

To address these limitations, we present 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 composite adaptive control, thereby enabling ground vehicles to navigate and adapt to complex terrains in real time. Our method is well-suited for any ground vehicle equipped with the following: 1) sensors to measure the internal robot 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 in real time;
- 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 and robustness against model errors for adaptive control with visual and state input that works in conjunction with our 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 and arbitrary time-varying disturbances.

We validate the effectiveness of our method both through simulation and in hardware on two heterogeneous robotic platforms, 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: Sec. II provides a review of existing literature. In Sec. III, our offline meta-learning algorithm and the online adaptation algorithm are presented. Sec. IV presents the two different vehicle models and the adaptive controllers with stability and robustness guarantees. In Sec. V, we analyze the VFM output for terrain, particularly in relation to our learning-based adaptive controller. In Sec. VI, we validate the algorithm using simulation and continue with experimental validation in Sec. VII and concluding remarks in Sec. 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 $(\mathbf{Ab})$ 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 $i, \ldots, j$. All matrices and vectors are written in bold.

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 from previously seen tasks to adapt rapidly to a new task [16], [17], [37], [38]. 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], [39]–[41] 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 the literature are Model-Agnostic Meta-Learning (MAML) [17] with its online extension [41], Meta-learning via Online Changepoint Analysis (MOCA) [42], and Domain Adversarially Invariant Meta-Learning (DAIML) used in Neural-Fly [27]. For task-centered datasets, MAML [17] 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 [42] proposes task unsegmented meta-learning via online changepoint analysis. DAIML [27] proposes an online meta-learning-based approach where a shared representation is learned offline (for example, using data from different wind conditions for a quadrotor), with a linear, low-dimensional part updated online through adaptive control.

Our method builds on the previous work [27] 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 are terrain independent but encapsulate the remaining disturbances not captured during training.

A. Embedding Visual Information in Classical Control and Reinforcement Learning

One of the early works on including visual information for control is visual servoing [43], a technique mainly used for robot manipulation. Recently, vision-based reinforcement learning (VRL) has demonstrated the capability to control agents in simulated environments [44], as well as robots in real environments, with applications to ground robots [45]–[47] and robot manipulation [48]–[50]. This capability is achieved by leveraging high-fidelity models in robotics simulators [51], [52], imitation learning from human demonstrations or techniques to bridge the sim-to-real gap of the learned policy [14]. Nevertheless, a 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 algorithms for interpreting terrain traversability as a reward map, thus enhancing the understanding of the environments [53], [54]. Despite progress in combining vision with Reinforcement Learning (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 incorporated 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-robot interaction 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], traversability estimation [54], and robot manipulation [60], [61]. One of their key advantages lies in the robustness against variations in lighting and occlusions [62], 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 [23]–[31], [63] is a control method with provable convergence guarantees in which a set of linear parameters is adapted online to compensate for disturbances at runtime. Typically, these linear parameters are multiplied by a basis function, which can be constant (as in the case of integral control), derived from physics [64], or represented using Radial Basis Functions (RBFs) [65] or DNNs [27]. First introduced in [23], [66], composite adaptation combines online parameter estimation and tracking-error adaptive control. A rigorous robustness/stability analysis for composite adaptation with a connection to deep meta-learning was first derived in [27] for flight control applications.

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 [67]–[72]. In [73], an adaptive energy-aware prediction and planning framework for vehicles navigating over terrains with varying and unknown properties was proposed and demonstrated in simulation. [74] 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 [75] develops a probabilistic traction model with uncertainty quantification using both semantic and geometric terrain features. In [76], a meta-learning-based approach to adapt probabilistic predictions of rover dynamics with Bayesian regression is used. While these approaches succeeded in developing different models for control and planning, incorporating a suitable model that fits the adaptive control framework is an open area of research.

In this paper, we establish an adaptive controller that can handle 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. This work can be viewed as a generalization and improvement of [27] with a new control matrix adaptation method using visual information and improved stability results. Our adaptive capability acts as an enhancement to the offline trained basis function, further improving tracking performance under challenging conditions.

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

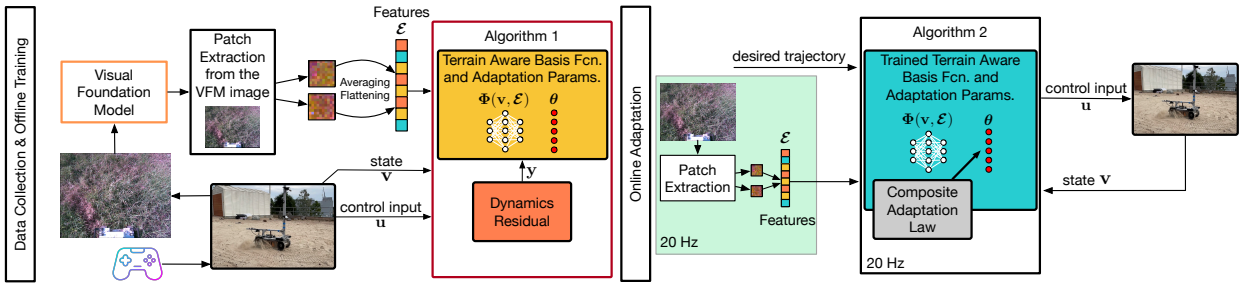


Fig. 2: Terrain-aware Architecture: offline data collection and training (Algorithm 1), followed by real-time adaptive control (Algorithm 2) running onboard the robot.

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 $\boldsymbol{\theta}(t) = [\theta_1 \dots \theta_{n_\theta}]^\top \in \mathbb{R}^{n_\theta}$ denotes a time-varying vector of linear parameters that are adapted online by our algorithm, $\boldsymbol{\delta} \in \mathbb{R}^n$ is a representation error, 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 the adaptive control part of our method, the precise form of $\boldsymbol{\mathcal{E}}$ is not required in our work. We can think of $\boldsymbol{\mathcal{E}}$ as computed by some arbitrary Lipschitz function from the robot's sensors to $\mathbb{R}^p$. We provide details on the particular form of $\boldsymbol{\mathcal{E}}$ used in our empirical sections (Sec. 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 adapt in real-time. The online adaptation is necessary in real scenarios, because two environments (i.e., terrains) might have the same representation $\boldsymbol{\mathcal{E}}$, but induce different dynamic behaviors onto the robot, as well as for other types of disturbances not captured in the feature mapping $\mathbf{\Phi}$.

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

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

where the dependence on a parameter vector $\mathbf{w}$ (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 by $\mathbf{\Phi}_i^{\mathbf{w}}(\mathbf{x}, \boldsymbol{\mathcal{E}}) : \mathbb{R}^{n+p} \mapsto \mathbb{R}^{n \times m}$. The control affine assumption is motivated by two factors. First, in our main application of ground vehicles with desired-velocity inputs, input-affine disturbances more accurately capture terrain interactions such as slippage (Sec. IV-C1), internal dynamics, and wheel or track degradations. Second, this assumption 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 disturbances by input-output stability combined with contraction theory [77], [78]. Lastly, we define the dynamics residual that is used in both the online and offline phase of our method as $\mathbf{y} = \dot{\mathbf{x}} - \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$.

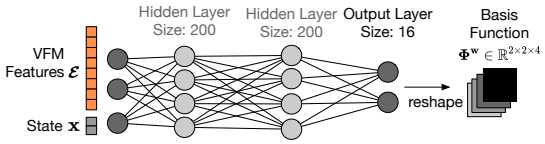


Fig. 3: The structure of the DNN used for the basis function $\mathbf{\Phi}^{\mathbf{w}}$ in the controller synthesis from (18), (19) applied to a tracked vehicle.

B. Offline Meta-Learning Phase

In Fig. 2, we illustrate the structure of our proposed solution to learn offline the basis function $\mathbf{\Phi}^{\mathbf{w}}(\mathbf{x}, \boldsymbol{\mathcal{E}})$ in (3) and to make real-time adjustments using composite adaptive control (Sec. 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 $\mathbf{\Phi}^{\mathbf{w}}$ in (3), we collect a dataset of the robot operating 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 representation $\boldsymbol{\mathcal{E}}$, as discussed in Sec. 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 representation, and residual dynamics derivative of the n^{th} trajectory.

2) Model Architecture: In designing the parameterized function class for the basis function $\mathbf{\Phi}^{\mathbf{w}}$, we prioritize simplicity and efficiency to enable fast inference for real-time control. Therefore, we select a fully connected DNN with two hidden layers (Fig. 3), which takes as input both the robot's state and visual features from the VFM. We employ layer-wise spectral

Algorithm 1 Offline Meta-Learning with Continuous Trajectories

Input
  Dataset of N trajectories of length ℓ, window length distribution L, regularization target $\boldsymbol{\theta}_{\mathrm{r}}$ and weight λ_{r}, minibatch size K, initial DNN weights $\mathbf{w}$.
Output: Final optimized DNN weights $\mathbf{w}$ of $\boldsymbol{\Phi}$.
while not converged **do**
  Sample (with replacement) size-K minibatches of:
    - trajectory indices $\{n_k\}_{k=1}^K$,
    - window lengths $\{\hat{\ell}_k\}_{k=1}^K$,
    - start times $\{s_k\}_{k=1}^K$.
  Solve (4) for each $\theta^\star_{n_k,\hat{\ell}_k,s_k}$ in the minibatch
    (in closed form, allowing $J_{n_k,\hat{\ell}_k,s_k}$ gradient flow[1]).
  Take optimizer step on $\mathbf{w}$ w.r.t minibatch cost (4)

$$\textstyle\sum_{k=1}^K J_{n_k,\hat{\ell}_k,s_k}(\mathbf{w}).$$

  Spectral Normalizaton: $\mathbf{W}_i \leftarrow \frac{\mathbf{W}_i}{\|\mathbf{W}_i\|}$ for all $i \in [d]$.
end while

Algorithm 2 Rapid Terrain-Informed Online Adaptation for Model Mismatch and Tracking Error

Input
  Optimized feature mapping $\boldsymbol{\Phi}$ from Algorithm 1, importance weight for prediction $\mathbf{R}$, damping constant λ, initial adaptive gain $\boldsymbol{\Gamma}_0$, reference trajectory $\mathbf{x}_r$.
Initialize $\hat{\boldsymbol{\theta}}$ with an user-defined regularization target $\boldsymbol{\theta}_{\mathrm{r}}$.
while running **do**
  Evaluate the VFM on the input image and get the features $\boldsymbol{\mathcal{E}}$.
  Get state $\mathbf{x}$ and evaluate the DNN $\boldsymbol{\Phi}(\mathbf{x}, \boldsymbol{\mathcal{E}})$.
  Compute the tracking error vector $\mathbf{s}$ (e.g., using (17)).
  Compute the control input $\mathbf{u}$ (e.g., using (18)).
  Compute the dynamics residual $\mathbf{y}$ (e.g., using (20))
  Compute the adaptation parameter derivative $\dot{\hat{\boldsymbol{\theta}}}$
  $\dot{\hat{\boldsymbol{\theta}}} = -\lambda\hat{\boldsymbol{\theta}} - \mathrm{predict}(\boldsymbol{\Gamma}, \mathbf{y}, \boldsymbol{\Phi}, \hat{\boldsymbol{\theta}}, \mathbf{u}) + \mathrm{track}(\boldsymbol{\Gamma}, \mathbf{s}, \boldsymbol{\Phi}, \hat{\boldsymbol{\theta}}, \mathbf{u})$.
  Compute the adaptation gain derivative $\dot{\boldsymbol{\Gamma}}$ (e.g., using (19) or (26)).
  Integrate with system timestep Δ_t

$$\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} + \Delta_t\dot{\hat{\boldsymbol{\theta}}}, \quad \boldsymbol{\Gamma} \leftarrow \boldsymbol{\Gamma} + \Delta_t\dot{\boldsymbol{\Gamma}}.$$

end while

normalization to constrain the Lipschitz constant of the DNN. Spectral normalization is crucial for ensuring smooth control outputs and limiting pathological behavior outside the training domain [79]. Details of spectral normalization are given in Sec. III-B3.

3) Optimization: Our method is built around the assumption that two terrains with similar visual features $\boldsymbol{\mathcal{E}}$ will usually, but not always, induce similar dynamics. We account for this observation 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 $\boldsymbol{\theta}$ 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 the weights $\mathbf{w}$ of the basis function 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 $n \in [1 \dots N]$, window length $\hat{\ell} \in [1 \dots \ell]$, and starting timestep $s \in [1 \dots \ell - \hat{\ell} + 1]$, the best-fit value

$$\begin{aligned}
&\operatorname*{argmin}_{\boldsymbol{\theta}} \sum_{t=s}^{s+\hat{\ell}-1} \left\|\mathbf{y}_t^n - (\boldsymbol{\Phi}^{\mathbf{w}}\boldsymbol{\theta})\mathbf{u}_t^n\right\|_2^2 + \lambda_r\|\boldsymbol{\theta} - \boldsymbol{\theta}_r\|_2^2, \\
&= \sum_{t=s}^{s+\hat{\ell}-1} \left((\boldsymbol{\Phi}^{\mathbf{w}}\boldsymbol{\theta})^\top(\boldsymbol{\Phi}^{\mathbf{w}}\boldsymbol{\theta}) + \lambda_r\mathbf{I}_{n_\theta}\right)^{-1}\left((\boldsymbol{\Phi}^{\mathbf{w}}\boldsymbol{\theta})^\top\mathbf{y}_t^n + \lambda_r\boldsymbol{\theta}_r\right) \\
&=: \boldsymbol{\theta}^\star_{n,\hat{\ell},s}(\mathbf{w}),
\end{aligned}$$

is the solution to an L_2-regularized linear least-squares problem and is a closed-form, continuous function of the feature mapping parameter $\mathbf{w}$, where $\boldsymbol{\Phi}^{\mathbf{w}}$ is shorthand for $\boldsymbol{\Phi}^{\mathbf{w}}(\mathbf{x}_t^n, \boldsymbol{\mathcal{E}}_t^n)$, $\lambda_r \in \mathbb{R}$ is the regularization parameter, and $\boldsymbol{\theta}_r \in \mathbb{R}^{n_\theta}$ is the regularization target, chosen arbitrary. The $\|\boldsymbol{\theta} - \boldsymbol{\theta}_r\|_2^2$ regularization term ensures that the closed-form solution is unique. 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

$$\begin{aligned}
J(\mathbf{w}) &= \mathop{\mathbb{E}}_{n,\hat{\ell},s}\left[\sum_{t=s}^{s+\hat{\ell}-1}\left\|\mathbf{y}_t^n - \left(\boldsymbol{\Phi}^{\mathbf{w}}(\mathbf{x}_t^n, \boldsymbol{\mathcal{E}}_t^n)\boldsymbol{\theta}^\star_{n,\hat{\ell},s}(\mathbf{w})\right)\mathbf{u}_t^n\right\|_2^2\right] \\
&:= \mathop{\mathbb{E}}_{n,\hat{\ell},s}\left[J_{n,\hat{\ell},s}(\mathbf{w})\right], \qquad (4)
\end{aligned}$$

where the expectation is shorthand for $n \sim \mathcal{U}[1, N]$, $\hat{\ell} \sim L$, and $s \sim \mathcal{U}[1, \ell - \hat{\ell} + 1]$. We incorporate the closed-form computation of the best-fit linear component $\boldsymbol{\theta}^\star_{n,\hat{\ell},s}$ in the computational graph of our optimization (meaning we take gradients through the least squares solution in Line 9), as opposed to treating the trajectories of $\boldsymbol{\theta}$ as optimization variables. Given this, 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 $\mathbf{w} = (\mathbf{W}_1, \dots, \mathbf{W}_d, \overline{\mathbf{w}})$, with d being the number of layers, where $\mathbf{W}_1, \dots, \mathbf{W}_d$ 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 $i \in [1 \dots d]$. This is implemented in Line 12 of Algorithm 1. Note that finding less conservative ways to enforce $\|\boldsymbol{\Phi}^{\mathbf{w}}\|_{\mathrm{Lip}} \leq 1$ for DNN is an active area of research.

For the remaining sections, the parameters $\mathbf{w}$ of the basis function $\boldsymbol{\Phi}$ are fixed. Therefore, we drop the superscript and

refer to $\mathbf{\Phi}^{\mathbf{w}}$ as $\mathbf{\Phi}$, for simplicity of notation.

C. Online $\boldsymbol{\theta}$ Adaptation and Tracking Control

This section introduces the online adaptation running onboard the robot to adapt the linear part $\boldsymbol{\theta}$ of the dynamics model. The algorithm uses composite adaptation [80] and is given in Algorithm 2. The parameter vector $\boldsymbol{\theta}$ is initialized with a user-defined regularization target $\boldsymbol{\theta}_{\mathrm{r}}$ (Line 3). 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 $\boldsymbol{\mathcal{E}}$ (Line 5). The feature mapping $\mathbf{\Phi}$ is then evaluated using the robot's current state $\mathbf{x}$ and the feature vector $\boldsymbol{\mathcal{E}}$ (Line 6). We compute the error vector $\mathbf{s}$ between the reference trajectory $\mathbf{x}_r$ and the actual state $\mathbf{x}$, for example, using (17), as well as the control input $\mathbf{u}$ (Line 8).

These previously computed variables are passed to the composite adaptive controller. In this way, model mismatches and other disturbances not captured during training (Algorithm 1) can be adapted in real-time. For each sampled measurement (interaction with the environment), the adaptation parameter vector $\hat{\boldsymbol{\theta}}$ is updated using the composite adaptation rule in Line (10), which is designed to decrease both the tracking error and the prediction error.

Each term of Line (10) provides a specific functionality: the first term in Line (10) 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 with respect to $\hat{\boldsymbol{\theta}}$, where $\mathbf{R}$ is a positive definite matrix. The third term minimizes the trajectory tracking error. In Line 11, we introduce $\mathbf{\Gamma}$, which is our adaptation gain, and its derivative $\dot{\mathbf{\Gamma}}$ can be defined from least-squares with exponential forgetting [64], [80] 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 functions 'predict' and 'track' are defined in Sec. IV-C3. The stability and robustness properties of Algorithm 2 are presented in Sec. IV-C.

IV. System Modelling and Control Synthesis

We apply the methods from Algorithm 1 and 2 to a skid-steering tracked vehicle (Fig. 4) (Sec. IV-A through IV-C) and to an Ackermann-steering vehicle (Sec. IV-D through IV-E). The 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 (10) with its corresponding simplified model of the form (11). To this simplified model, we apply an adaptive controller with learned ground information of the form (18) and (19). The car-like vehicle uses the Ackermann steering geometry, which ensures that all wheels turn around the same center, thus minimizing wheel wear. For this vehicle type, we derive a 3-DOF dynamics model (28) using the bicycle model and an adaptive controller using Algorithm 2.

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

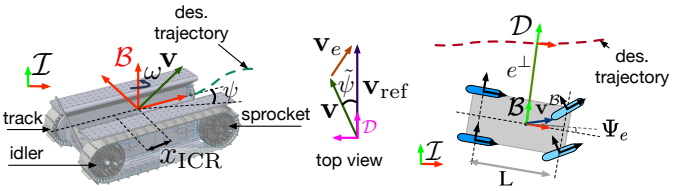


Fig. 4: The frames of reference for the tracked vehicle, its corresponding velocities, and the main driving components (left), a velocity vector diagram used for the proof of Theorem 2 (middle), and the car model notations (right). For both vehicles, we assume the center of mass and the body frame are at the same location.

A. Tracked Vehicle Dynamics Model

We define 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. Consider the 3-DOF 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

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{B}(\mathbf{q})\boldsymbol{\tau}_u + \mathbf{F}_r(\mathbf{q},\dot{\mathbf{q}}), \tag{5}$$

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 Coriolis 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 (5) is difficult due to underactuation. To address this complexity, previous work [81], [82] introduced a nonholonomic constraint for (5), which reduces the number of state variables. The following constrains the ratio of the lateral body velocity $\dot{p}_y^{\mathcal{B}}$ to the angular velocity ω

$$\dot{p}_y^{\mathcal{B}} + x_{\mathrm{ICR}}\omega = 0, \tag{6}$$

where x_{ICR} is the Instantaneous Center of Rotation (ICR) and $\omega = \dot{\psi}$. We embed this constraint into (5), as follows

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

with λ_c being the Lagrange multiplier corresponding to the equality constraint in (6). By assuming x_{ICR} as constant, $\mathbf{A}(\mathbf{q}) \in \mathbb{R}^{1\times3}$ is defined as follows, in which $\mathbf{p}$ from (6) is expressed in the $\mathcal{I}$ frame

$$\begin{bmatrix}-\sin\psi & \cos\psi & x_{\mathrm{ICR}}\end{bmatrix} \cdot \begin{bmatrix}\dot{p}_x^{\mathcal{I}} & \dot{p}_y^{\mathcal{I}} & \omega\end{bmatrix} = \mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = 0. \tag{8}$$

To remove the constraint force from (7), 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 $\mathbf{S}(\mathbf{q})^\top\mathbf{A}(\mathbf{q})^\top = \mathbf{0}$ [83], [84].

$$\mathbf{S}(\mathbf{q}) = \begin{bmatrix}\cos(\psi) & x_{\mathrm{ICR}}\sin(\psi)\\ \sin(\psi) & -x_{\mathrm{ICR}}\cos(\psi)\\ 0 & 1\end{bmatrix}. \tag{9}$$

We select 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 [85]

$$\begin{aligned}\dot{\mathbf{q}}(t) &= \mathbf{S}(\mathbf{q})\mathbf{v}(t),\\ \dot{\mathbf{v}}(t) &= \widetilde{\mathbf{M}}^{-1}(\widetilde{\mathbf{B}}(\mathbf{q})\boldsymbol{\tau}_u - \widetilde{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})\mathbf{v}(t) + \widetilde{\mathbf{F}}_r(\mathbf{q},\dot{\mathbf{q}})),\end{aligned} \tag{10}$$

with the reduced matrices

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

where m is the mass of the robot and I_z is the inertia of the robot about the rotational degree of freedom.

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 inputs. Consequently, we have chosen to simplify the system in (10) with the velocity modeled as a first-order time delay

$$\dot{\mathbf{q}}(t) = \mathbf{S}(\mathbf{q})\mathbf{v}(t), \quad \dot{\mathbf{v}}(t) = \mathbf{A}_{\mathrm{n}}\mathbf{v}(t) + \mathbf{B}_{\mathrm{n}}\mathbf{u}(t), \tag{11}$$

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

$$\mathbf{A}_{\mathrm{n}} = \begin{bmatrix} -\frac{1}{\tau_v} & 0 \\ 0 & -\frac{1}{\tau_\omega} \end{bmatrix}, \quad \mathbf{B}_{\mathrm{n}} = \begin{bmatrix} \frac{k_1}{\tau_v} & 0 \\ 0 & \frac{k_2}{\tau_\omega} \end{bmatrix}. \tag{12}$$

The simplified system is dynamically equivalent to (10). We identify the process gains k_1, k_2 and the process time constants τ_v, τ_ω using system identification on hardware. The robot is symmetric and rotates around the origin, therefore x_{ICR} is assumed 0.

C. Adaptive Tracking Controller for a Tracked Vehicle

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 (11) and prove its exponential convergence to a bounded error ball. Note that our adaptive controller can be applied to any system of the form (5).

1) Motivation for Control Matrix Adaptation: The longitudinal slip κ is defined [86] as

$$\kappa = -\frac{v_x^{\mathcal{B}} - \Omega_{\mathrm{tr}} r_{\mathrm{tr}}}{v_x^{\mathcal{B}}}, \tag{13}$$

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 (13) 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 $\kappa \to \infty$, 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

$$\mathbf{v}_{\mathrm{ref}}^{\mathcal{I}} = \mathbf{v}_d^{\mathcal{I}} - \mathbf{K}_p\tilde{\mathbf{p}}^{\mathcal{I}}, \quad v_{\mathrm{ref,x}} = \begin{bmatrix}\cos(\psi)\\ \sin(\psi)\end{bmatrix} \cdot \mathbf{v}_{\mathrm{ref}}^{\mathcal{I}}, \tag{14}$$

$$\omega_{\mathrm{ref}} = \dot{\psi}_{\mathrm{ref}} - k_\psi(\psi - \psi_{\mathrm{ref}}), \tag{15}$$

where the reference angle is given as

$$\psi_{\mathrm{ref}} = \begin{cases} \arctan\left(\frac{\mathbf{v}_{\mathrm{ref,y}}^{\mathcal{I}}}{\mathbf{v}_{\mathrm{ref,x}}^{\mathcal{I}}}\right), & \text{if } \|\mathbf{v}_{\mathrm{ref}}^{\mathcal{I}}\|_2^2 > v_\epsilon \\ \psi_{\mathrm{d}}, & \text{otherwise.} \end{cases} \tag{16}$$

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 further used in the control synthesis.

3) Controller Synthesis: We design a composite adaptive controller $\mathbf{u}(t)$ and show that this composite tracking and adaptation error exponentially converges to a bounded error ball. First, we start by defining the tracking error variable $\mathbf{s}$ as

$$\mathbf{s} = \mathbf{v} - \mathbf{v}_{\mathrm{ref}} = [v_x^{\mathcal{B}} - v_{\mathrm{ref,x}},\ \omega - \omega_{\mathrm{ref}}]^\top. \tag{17}$$

We then derive the tracking controller for the system in (11)

$$\mathbf{u} = -(\mathbf{B}_{\mathrm{n}} + \boldsymbol{\Phi}\hat{\boldsymbol{\theta}})^{-1}[\mathbf{K}\mathbf{s} + \mathbf{A}_{\mathrm{n}}\mathbf{v}_{\mathrm{ref}} - \dot{\mathbf{v}}_{\mathrm{ref}}], \tag{18}$$

where $\mathbf{K} \in \mathbb{R}^{2\times 2}$ is a positive gain matrix, with $\mathbf{K} = \mathrm{diag}(k_{dx}, k_{dw})$, $\boldsymbol{\Phi} \in \mathbb{R}^{2\times 2\times n_\theta}$ is the output of the DNN basis function (Fig. 3) evaluated with the feature vector $\boldsymbol{\mathcal{E}}$ and state $\mathbf{v}$, and $\hat{\boldsymbol{\theta}} \in \mathbb{R}^{n_\theta}$ is the estimated parameter vector of the true parameter vector $\boldsymbol{\theta}$. Recall from Sec. III-A that the learned basis function $\boldsymbol{\Phi}$ and the true adaptation parameters $\boldsymbol{\theta}$ were introduced to model the disturbance $\mathbf{d} \approx (\boldsymbol{\Phi}\boldsymbol{\theta})\mathbf{u} = \sum_{i=1}^{n_\theta} \theta_i \boldsymbol{\Phi}\mathbf{u}$. For our model of the skid-steer vehicle, we chose $n_\theta = 4$, matching the number of terms in our control matrix $\mathbf{B}_{\mathrm{n}}$. Choosing n_θ too large can introduce redundant parameters and choosing n_θ too small could make the function class insufficiently expressive.

Theorem 1. *By applying the controller in (18) to the dynamics that evolve according to (11), with the composite adaptation law, for each $i \in [1, n_\theta]$*

$$\dot{\hat{\theta}}_i = -\lambda\hat{\theta}_i - \underbrace{\gamma_i \mathbf{u}^\top \mathbf{\Phi}_i^\top \mathbf{R}^{-1}\left(\sum_{i=1}^{n_\theta}\mathbf{\Phi}_i\mathbf{u}\hat{\theta}_i - \mathbf{y}\right)}_{\text{predict}} + \underbrace{\gamma_i \mathbf{s}^\top \mathbf{\Phi}_i \mathbf{u}}_{\text{track}},$$
$$\dot{\gamma}_i = -2\lambda\gamma_i + q_i + \gamma_i \mathbf{u}^\top \mathbf{\Phi}_i^\top \mathbf{R}^{-1}\mathbf{\Phi}_i\mathbf{u}\gamma_i, \tag{19}$$

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 $\mathbf{B} := \mathbf{B}_\mathrm{n} + \mathbf{\Phi}\boldsymbol{\theta}$, we obtain the following closed loop system using (11)

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

where $\boldsymbol{\delta}$ is a representation error, previously introduced in (2). Let $\tilde{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}$ be the error adaptation vector. Further, using the composite variable $\mathbf{s}$ in (17), the closed-loop system becomes

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

For the prediction term in (19), we compute the dynamics residual derivative $\mathbf{y}$ determined for the bounded and adversarial noise $\bar{\boldsymbol{\epsilon}}$ as

$$\mathbf{y} = \mathrm{LPF}(\mathrm{s})\dot{\mathbf{v}} - \mathbf{f}(\mathbf{v}, \mathbf{u}, t) = (\mathbf{\Phi}\boldsymbol{\theta})\mathbf{u} + \bar{\boldsymbol{\epsilon}}, \tag{20}$$

where premultiplying the noisy measurement $\dot{\mathbf{v}}$ by $\mathrm{LPF}(s)$ with the Laplace transform variable s indicates low-pass filtering. Using the Lyapunov function $\mathcal{V} = \mathbf{s}^\top\mathbf{s} + \sum_{i=1}^{n_\theta}\tilde{\theta}_i\gamma_i^{-1}\tilde{\theta}_i$, we compute its derivative as follows

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

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

$$\begin{aligned}
\dot{\mathcal{V}} &= -2\mathbf{s}^\top(\mathbf{K} - \mathbf{A}_\mathrm{n})\mathbf{s} - \sum_{i=1}^{n_\theta}\tilde{\theta}_i(q_i\gamma_i^{-2} + \mathbf{u}^\top\mathbf{\Phi}_i^\top\mathbf{R}^{-1}\mathbf{\Phi}_i\mathbf{u})\tilde{\theta}_i \\
&- 2\left(\sum_{i=1}^{n_\theta}\tilde{\theta}_i\mathbf{u}^\top\mathbf{\Phi}_i^\top\right)\mathbf{R}^{-1}\left(\sum_{j=1}^{n_\theta}\mathbf{\Phi}_j\mathbf{u}\tilde{\theta}_j\right) + \text{error terms}.
\end{aligned} \tag{21}$$

Next, we will bound the terms in (21) as follows. There exists $\alpha \in \mathbb{R}_+$ such that

$$\begin{aligned}
-2(\mathbf{K} - \mathbf{A}_\mathrm{n}) &\preceq -2\alpha\mathbf{I}, \\
-\left(q_i\gamma_i^{-2} + \mathbf{u}^\top\mathbf{\Phi}_i^\top\mathbf{R}^{-1}\mathbf{\Phi}_i\mathbf{u}\right) &\leq -2\alpha\gamma_i^{-1}, \quad \forall i \in [1, n_\theta].
\end{aligned} \tag{22}$$

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

$$\bar{d} = \sup_t\left(\|\boldsymbol{\delta}\| + \left|\sum_{i=1}^{n_\theta}\left(\mathbf{u}^\top\mathbf{\Phi}_i^\top\mathbf{R}^{-1}\bar{\boldsymbol{\epsilon}} - \gamma_i^{-1}\lambda\theta_i - \gamma_i^{-1}\dot{\theta}_i\right)\right|\right), \tag{23}$$

Note that this is a conservative estimate (the worst-case disturbance of all future time t), and hence can be made smaller using a shorter time range. Furthermore, even for a relatively large value of $\mathbf{\Phi}$, $\bar{d}$ can be made small using a larger value of $\mathbf{R}$ and a smaller value of $\bar{\boldsymbol{\epsilon}}$. We define the matrix $\mathcal{M}$, for $i \in [1, n_\theta]$

$$\mathcal{M} = \begin{bmatrix}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathrm{diag}(\gamma_i^{-1})\end{bmatrix}. \tag{24}$$

By applying the Comparison Lemma [87] and using a contraction theory like argument [78], [88], we can then prove the tracking error and adaptation parameters error exponentially converge to the bounded error ball

$$\lim_{t\to\infty}\left\|\begin{bmatrix}\mathbf{s} \\ \tilde{\theta}_i\end{bmatrix}\right\| \leq \frac{\bar{d}}{\alpha\lambda_{\min}(\mathcal{M})} := \bar{b}, \tag{25}$$

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

It follows from [77], [87] that the input-to-state stability (ISS) and bounded input and bounded output (BIBO) stability in the sense of finite-gain $\mathcal{L}_p$ [87] is proven for $\bar{b} \in \mathcal{L}_{pe}$, 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 $0 \leq t \leq \tau$. □

The exponential convergence proof in Theorem 1 shows that the online algorithm (Algorithm 2) will drive $\hat{\boldsymbol{\theta}}$ to a value within a bounded error ball of the offline least-squares solution used in the meta-learning algorithm (Algorithm 1) for a sufficiently long window of data. In contrast with [27], [80], (18) and (19) admits adaptation through the $\mathbf{B}$ control influence matrix. For stability purposes, under the assumption of a diagonal $\mathbf{\Gamma}$, the adaptation law equation resembles the Riccati equation of the $\mathcal{H}_\infty$ filtering [89]. This tends to increase the adaptation gain, making it more responsive to measurements.

The parameters of the adaptation law (19) are $\mathbf{\Gamma} = \mathrm{diag}(\gamma_1, \ldots, \gamma_{n_\theta})$, $\mathbf{R}$, λ, and $\mathbf{Q} = \mathrm{diag}(q_i)$. $\mathbf{\Gamma}$ is a positive definite matrix that influences the convergence rate of the

estimator, and a sufficiently large initial $\mathbf{\Gamma}_0$ 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, and $\mathbf{R}$ 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.

Next, we assume the adaptation gain Γ in (19) has cross terms. Under this more general setting, we prove the exponential convergence of both $\tilde{\boldsymbol{\theta}}$ and $\mathbf{s}$ to a bounded error ball.

Proposition 1. *By applying the controller in (18) to the dynamics in (11), with the composite adaptation law*

$$\dot{\hat{\boldsymbol{\theta}}} = -\lambda\hat{\boldsymbol{\theta}} - \underbrace{\mathbf{\Gamma}\mathbf{H}^\top\mathbf{R}^{-1}(\mathbf{H}\hat{\boldsymbol{\theta}} - \mathbf{y})}_{\text{predict}} + \underbrace{\mathbf{\Gamma}\mathbf{H}^\top\mathbf{s}}_{\text{track}}, \tag{26a}$$

$$\dot{\mathbf{\Gamma}} = -2\lambda\mathbf{\Gamma} + \mathbf{Q} - \mathbf{\Gamma}\mathbf{H}^\top\mathbf{R}^{-1}\mathbf{H}\mathbf{\Gamma}, \tag{26b}$$

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

Proof. We define the matrix $\mathbf{H} = [\mathbf{h}_1 \dots \mathbf{h}_{n_\theta}] \in \mathbb{R}^{n\times n_\theta}$, where the columns $\mathbf{h}_i = \mathbf{\Phi}_i\mathbf{u}$, for each $i \in [1, n_\theta]$. By observing that the disturbance in (3) can be defined as $\mathbf{d} \approx \sum_{i=1}^{n_\theta} \mathbf{\Phi}_i\theta_i\mathbf{u} := \mathbf{H}\boldsymbol{\theta}$, the proof of exponential convergence for additive disturbance adaptation from [27] can be directly applicable for the multiplicative disturbance adaptation. □

Note that the exponential convergence proof for Proposition 1 using Lyapunov theory holds for both when the last term of the gain adaptation law (26b) is positive and when the last term is negative. A negative sign makes the update law (26b) resemble the covariance update law of the Kalman filter. However, using a positive sign will make the closed-loop system converge faster for the same constants. Our controller in Theorem 1 behaves similar to the second case with the assumption that $\mathbf{\Gamma}$ is diagonal.

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 (25) 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 (15), 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 $\mathbf{v} = \mathbf{v}_{\text{ref}} + \mathbf{v}_e$ in vector form, 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

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

We compute and bound the norm, as follows

$$\|\mathbf{v}_e^{\mathcal{D}}\|_2 \leq v_{\max} \left| \sin\left(\frac{\bar{b}}{k_\psi}\right) \right| \sqrt{2} + \bar{b}, \tag{27}$$

where $v_{\max}$ is our assumption for the existence of an upper bound for the reference velocity. From (14) and (27), it is straightforward to see that the position error is also bounded using the Comparison Lemma. □

D. Ackermann Steering Vehicle Dynamics Model

We define a fixed reference frame $\mathcal{I}$, a moving reference frame $\mathcal{B}$ attached to the center of mass of the car, and a desired frame $\mathcal{D}$ attached to the desired trajectory as seen in Fig. 4. Similar to (8), a non-holonomic constraint holds: $\begin{bmatrix} \sin\psi & -\cos\psi & 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{p}_x^{\mathcal{I}} & \dot{p}_y^{\mathcal{I}} & 0 \end{bmatrix}$, where $\dot{p}_x^{\mathcal{I}}$ and $\dot{p}_y^{\mathcal{I}}$ are the velocities in the inertial frame $\mathcal{I}$ and ψ is the yaw angle from $\mathcal{B}$ to $\mathcal{I}$. For the tracked vehicle discussed in Sec. IV-A, the instantaneous center of rotation x_{ICR} is assumed to be 0 with $\dot{p}_y^{\mathcal{B}} = 0$ in (6) because the tracked vehicle is not designed for highly aggressive maneuvers.

A car, on the other hand, can be drifting, and thus the side velocity plays a much more important role, which is considered in our control design. Let $v_x^{\mathcal{B}}$ and $v_y^{\mathcal{B}}$ be the linear velocities in the body frame and ω the angular velocity around the vertical z-axis of the $\mathcal{B}$ frame. The dynamic model can be expressed as [90]

$$\begin{aligned} m(\dot{v}_x^{\mathcal{B}} - \omega v_y^{\mathcal{B}}) &= F_{xr} + F_{xf}\cos(u_\delta) - F_{yf}\sin(u_\delta), \\ m(\dot{v}_y^{\mathcal{B}} + \omega v_x^{\mathcal{B}}) &= F_{yr} + F_{xf}\sin(u_\delta) + F_{yf}\cos(u_\delta), \\ I_z\dot{\omega} &= \frac{L}{2}F_{xf}\sin(u_\delta) + \frac{L}{2}F_{yf}\cos(u_\delta) - \frac{L}{2}F_{yr}, \end{aligned} \tag{28}$$

where L is the wheelbase length, F_{xf} and F_{xr} are the front and rear tire forward forces, m is the vehicle mass, I_z is the vehicle inertia about the vertical axis intersecting the center of mass, and the lateral forces are $F_{yf} \approx C_y\alpha_f$, $F_{yr} \approx C_y\alpha_r$, where C_y is the tire cornering stiffness [11], and α_r and α_f are two tire slip angles, defined as in [90]. The tire cornering stiffness coefficient is terrain- and wheel-dependent. Either an accurate estimate or online adaptation is necessary when designing a tracking controller. Note that a more slippery ground has a lower C_y.

We decouple the controller for the longitudinal velocity from the controller for the lateral and angular velocity and apply our MAGIC$^{\text{VFM}}$ algorithm to the lateral and angular motion. Note that the forward velocity dynamics is nonlinear. Therefore, for simplicity, similar to the tracked vehicle, we model the forward velocity as a first-order time delay system $\dot{v}_x^{\mathcal{B}} = -\tau_v^{-1}(v_x^{\mathcal{B}} - u_v)$, with the time constant τ_v identified through system identification. We then design an exponentially stabilizing PD tracking controller for this linear system. For the lateral and angular motion, we linearize (28) around a zero steering angle and assume a small tire slip angle. The resulting linear time-varying system dynamics in the $\mathcal{B}$ frame is written by using $\mathbf{x} := [v_y^{\mathcal{B}}, \omega]$ and the disturbance model (3)

$$\dot{\mathbf{x}} = \mathbf{A}_{\text{n}}(v_x^{\mathcal{B}}(t))\mathbf{x} + \mathbf{B}_{\text{n}}u_\delta + (\mathbf{\Phi}(\mathbf{x}, \boldsymbol{\mathcal{E}})\boldsymbol{\theta})\, u_\delta, \tag{29}$$

where

$$\mathbf{A}_{\mathrm{n}}(v_x^{\mathcal{B}}(t)) = \begin{bmatrix} -\frac{2C_y}{mv_x^{\mathcal{B}}} & -v_x^{\mathcal{B}} \\ 0 & -\frac{L^2C_y}{2v_x^{\mathcal{B}}I_z} \end{bmatrix}, \mathbf{B}_{\mathrm{n}} = \begin{bmatrix} \frac{C_y}{m} \\ \frac{LC_y}{2I_z} \end{bmatrix},$$

and $\boldsymbol{\Phi}(\mathbf{x}, \boldsymbol{\mathcal{E}}) = \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 \end{bmatrix}^\top$, with the estimated component $\hat{\boldsymbol{\theta}}$ adapted online. The definition of $\boldsymbol{\Phi}(\mathbf{x}, \boldsymbol{\mathcal{E}}) \in \mathbb{R}^{2\times n_\theta}$ and $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ are the same as their definitions for the tracked vehicle. The adaptation component accounts for model mismatches as well as for the linearization errors in (28).

E. Adaptive Tracking Controller for Ackermann Steering

We apply MAGIC$^{\text{VFM}}$ to compensate for the sideslip when the robot is performing fast turning maneuvers. Thus, our adaptive control algorithm is applied only to the lateral and angular controller, although it can be applied for the linear velocity, as well. We define the path error $\mathbf{e} = [e^{\parallel}, e^{\perp}]$ with the longitudinal and lateral error components, as seen in Fig. 4

$$\mathbf{e} := \mathbf{p}^{\mathcal{D}} - \mathbf{p}_d^{\mathcal{D}} = \mathbf{R}_{\mathcal{I}}^{\mathcal{D}}(\mathbf{p}^{\mathcal{I}} - \mathbf{O}_{\mathcal{D}}^{\mathcal{I}}) \tag{30}$$

where $\mathbf{O}_{\mathcal{D}}^{\mathcal{I}}$ is the origin of the desired frame $\mathcal{D}$ expressed in $\mathcal{I}$, $\mathbf{p} = [p_x, p_y]$ is the position of the robot, $\mathbf{p}_d = [p_{x,d}, p_{y,d}]$ is the desired position from the trajectory, and $\mathbf{R}_{\mathcal{I}}^{\mathcal{D}}$ is the rotation from the inertial frame $\mathcal{I}$ to the desired frame $\mathcal{D}$. Next, we compute the time derivative of the path error (30) as follows

$$\begin{bmatrix} \dot{e}^{\parallel} \\ \dot{e}^{\perp} \end{bmatrix} = \mathbf{R}_{\mathcal{I}}^{\mathcal{D}}\mathbf{R}_{\mathcal{B}}^{\mathcal{I}}\mathbf{v}^{\mathcal{B}} - \mathbf{v}_d^{\mathcal{D}} + \dot{\mathbf{R}}_{\mathcal{I}}^{\mathcal{D}}(\mathbf{R}_{\mathcal{D}}^{\mathcal{I}}\mathbf{p}^{\mathcal{D}}), \tag{31}$$

where $\mathbf{R}_{\mathcal{B}}^{\mathcal{I}}$ is the rotation from the body frame $\mathcal{B}$ to the inertial frame $\mathcal{I}$, $\mathbf{v}_d^{\mathcal{D}} = [v_{d,x}^{\mathcal{D}}, 0]$ is the derivative of the desired position taken in $\mathcal{I}$, and expressed in $\mathcal{D}$, and $\mathbf{v}^{\mathcal{B}} = [v_x^{\mathcal{B}}, v_y^{\mathcal{B}}]$ is the velocity of the robot in the $\mathcal{B}$ frame. From (31), the perpendicular error derivative becomes

$$\dot{e}^{\perp} = \begin{bmatrix} \sin(\psi_e) \\ \cos(\psi_e) \end{bmatrix} \cdot \mathbf{v}^{\mathcal{B}} - \dot{\psi}_d e^{\parallel}, \tag{32}$$

where $\psi_e = \psi - \psi_d$ is the angle error between the actual orientation and the desired orientation. In addition, each component in (30) is

$$e^{\perp} = \begin{bmatrix} \sin(\psi_e) \\ \cos(\psi_e) \end{bmatrix} \cdot \tilde{\mathbf{p}}^{\mathcal{B}}, \quad e^{\parallel} = \begin{bmatrix} \cos(\psi_e) \\ -\sin(\psi_e) \end{bmatrix} \cdot \tilde{\mathbf{p}}^{\mathcal{B}}, \tag{33}$$

where $\tilde{\mathbf{p}}^{\mathcal{B}} = \mathbf{p}^{\mathcal{B}} - \mathbf{p}_d^{\mathcal{B}}$ is the position error expressed in $\mathcal{B}$. Because we model the dynamics decoupled and linearized, from (32), we obtain

$$\dot{e}^{\perp} = v_y^{\mathcal{B}} + v_x^{\mathcal{B}}\psi_e. \tag{34}$$

Further, we differentiate (34) and substitute (29), as follows

$$\ddot{e}^{\perp} = -\frac{2C_y}{mv_x^{\mathcal{B}}}v_y^{\mathcal{B}} - v_x^{\mathcal{B}}\omega + \frac{C_y}{m}u_\delta + (\boldsymbol{\phi}_1\boldsymbol{\theta})\, u_\delta + \dot{v}_x^{\mathcal{B}}\psi_e + v_x^{\mathcal{B}}\omega_e. \tag{35}$$

Now we design a tracking controller for the lateral motion of the vehicle. Let $s^{\perp} = \dot{e}^{\perp} + k_p e^{\perp}$, with $k_p \in \mathbb{R}^+$ a positive constant. Then, using (35), $\dot{s}^{\perp}$ is

$$\dot{s}^{\perp} = -\frac{2C_y}{mv_x^{\mathcal{B}}}v_y^{\mathcal{B}} - v_x^{\mathcal{B}}\omega + \frac{C_y}{m}u_\delta + (\boldsymbol{\phi}_1\boldsymbol{\theta})\, u_\delta + \dot{v}_x^{\mathcal{B}}\psi_e + v_x^{\mathcal{B}}\omega_e + k_p\dot{e}^{\perp}.$$

We then design the following adaptive controller

$$u_\delta = -\hat{b}_n^{-1}\left(k_v s^{\perp} - \frac{2C_y}{mv_x^{\mathcal{B}}}v_y^{\mathcal{B}} + \dot{v}_x^{\mathcal{B}}\psi_e - v_x^{\mathcal{B}}\omega_d + k_p\dot{e}^{\perp}\right), \tag{36}$$

where $\hat{b}_n = \frac{C_y}{m} + \boldsymbol{\phi}_1\hat{\boldsymbol{\theta}}$. Letting $\tilde{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}$, the closed-loop system of $s^{\perp}$ becomes

$$\dot{s}^{\perp} + k_v s^{\perp} = \left(\boldsymbol{\phi}_1\tilde{\boldsymbol{\theta}}\right) u_\delta. \tag{37}$$

Note that the controller in (36) resembles (18), which was derived for the tracked vehicle. Using the same proof as in Sec. IV-C3, we show that the tracking error $s^{\perp}$ and $\tilde{\boldsymbol{\theta}}$ exponentially converge to a bounded error ball. Next, we analyze the stability of the internal states $v_y^{\mathcal{B}}(t)$ and $\psi_e(t)$ under the exact error definitions (32)-(33).

Theorem 3. *If $|s^{\perp}(t)| \leq e^{-\gamma t}|s_0^{\perp}| + \frac{\epsilon}{\gamma}$, for positive constants γ and ϵ, and s_0 being the initial value, under the local assumption of $-\frac{\pi}{2} < \psi_e < \frac{\pi}{2}$ and a positive $v_x^{\mathcal{B}}$, then $\tilde{p}_y^{\mathcal{B}}$, $v_y^{\mathcal{B}}$, and ψ_e exponentially tend to bounds.*

Proof. Our forward velocity controller ensures $v_x^{\mathcal{B}}$ converges to the desired forward velocity, as shown in Theorem 1 for the tracked vehicle. Thus, we can approximate $\tilde{p}_x^{\mathcal{B}} \approx 0$ in (33). Hence, (32) and (33) are simplified as

$$e^{\perp} \approx \cos\psi_e\tilde{p}_y^{\mathcal{B}}, \quad \dot{e}^{\perp} \approx \sin\psi_e(v_x^{\mathcal{B}} + \dot{\psi}_d\tilde{p}_y^{\mathcal{B}}) + \cos\psi_e v_y^{\mathcal{B}}. \tag{38}$$

Note that under no disturbance ($\epsilon \approx 0$) and since $s^{\perp} = \dot{e}^{\perp} + k_p e^{\perp}$, $e^{\perp}$ and $\dot{e}^{\perp}$ exponentially converge to 0. Assuming a feasible reference trajectory (nonzero desired side velocity, $v_{y,d}^{\mathcal{B}}$), since $\cos\psi_e$ and $v_x^{\mathcal{B}}$ are nonzero values, we have $\tilde{p}_y^{\mathcal{B}}$, $\tilde{v}_y^{\mathcal{B}}$, and ψ_e converge to 0. If ϵ is a small nonzero value, assuming $\inf_t(\cos\psi_e) = \bar{\psi}$, we can show that $|\tilde{p}_y^{\mathcal{B}}|$ exponentially converges to a small error bound, as follows

$$\lim_{t\to\infty}|\tilde{p}_y^{\mathcal{B}}| \leq \frac{\epsilon}{\bar{\psi}k_p\gamma}. \tag{39}$$

We further assume that $|\tilde{p}_y^{\mathcal{B}}(t)| \approx e^{-\gamma_p t}|\tilde{p}_y^{\mathcal{B}}(0)| + \frac{|d(t)|}{k_p\gamma}$, where γ_p is a positive constant and $d(t)$ is a function with a small Lipschitz constant ϵ_y. With this assumption, we can show $v_y^{\mathcal{B}}$ exponentially converges to the bound $\frac{\epsilon_y}{k_p\gamma}$. Then, assuming $\inf_t v_x^{\mathcal{B}} = \bar{v}$, with positive $\bar{v}$, we apply the triangle inequality for $\dot{e}^{\perp}$ as follows

$$|\sin\psi_e(v_x^{\mathcal{B}} + \dot{\psi}_d\tilde{p}_y^{\mathcal{B}})| \leq |\dot{e}^{\perp}| + |\cos\psi_e v_y^{\mathcal{B}}|. \tag{40}$$

Taking the limit and denoting $\bar{v} - \frac{\epsilon\sup_t|\dot{\psi}_d|}{\bar{\psi}k_p\gamma}$ as $\underline{v}$, we show that ψ_e exponentially converges to a bounded error as

$$\lim_{t\to\infty}|\psi_e| \leq \arcsin\left(\frac{2\epsilon}{\underline{v}\gamma} + \frac{\bar{\psi}\epsilon_y}{\underline{v}k_p\gamma}\right). \tag{41}$$

Note that γ and k_p can be chosen to make the error bounds sufficiently small. The proof above is based on (32). Using the simplified version of (34), the bound simplifies to $\lim_{t\to\infty}|\psi_e| \leq \frac{2\epsilon}{\underline{v}\gamma} + \frac{\epsilon_y}{\underline{v}k_p\gamma}$. □

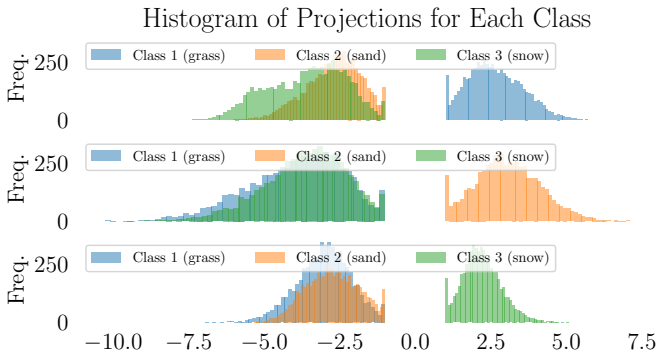


Fig. 5: DINO VFM discriminative ability for different terrains. We show the histograms of the projection values onto the separating hyperplane normal computed using Support Vector Classifier for 3 sets of classes with 5 images each (each row presents the separation margin between one class type and the other 2 classes). Note that the spikes at -1 and 1 are an artifact of the high dimensionality and the small dataset we used.

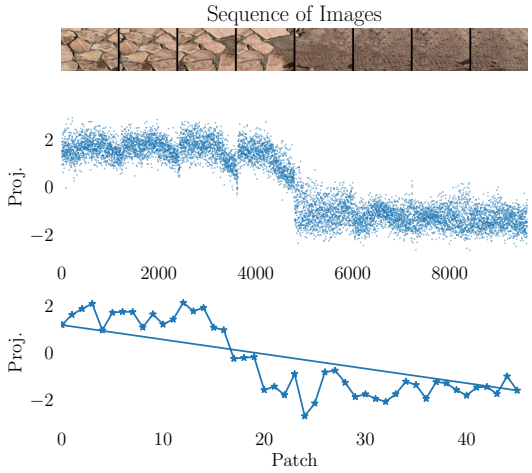


Fig. 6: Projection of sequential flagstones and gravel features onto an OVR-SVC separating hyperplane normal. The middle plot shows the projection of all the patch features from the top 8 figures, while the bottom plot shows the projection of a central patch taken from 45 sequential images of flagstones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

V. Empirical Results: Analysis of VFM Suitability

For our empirical work, we selected DINO V1 [91] 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

$$\mathcal{I}_{\text{RGB}} : h \times w \times 3 \to \mathcal{I}_{\text{VFM}} : \begin{bmatrix} h \\ \xi \end{bmatrix} \times \begin{bmatrix} w \\ \xi \end{bmatrix} \times \xi_f, \qquad (42)$$

where h and w are the image height and width. We then extract prominent patches from $\mathcal{I}_{\text{VFM}}$ to form the terrain representation $\boldsymbol{\mathcal{E}}$, which will be further used in the $\boldsymbol{\Phi}$ from (3). For our experiments, we select a set of patches on right and left of the tracks/wheels of the vehicle, as emphasized in Fig. 1.

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 ImageNet dataset, which also includes diverse ground terrains but mainly in the context of buildings, plants or landscapes, 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 [92]. We then project the feature vectors onto the one-vs-rest separating hyperplane normals. Let the separating hyperplane have the equation $\mathbf{w}_h \cdot \mathbf{x} + b_h = 0$, 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.

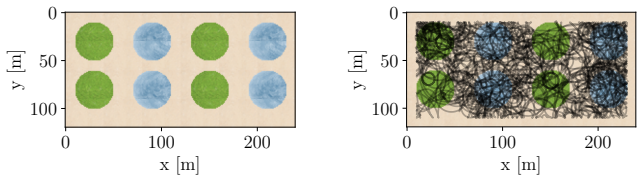
We next examine the distribution of the features across a sequence of images, taken while navigating from flagstones (irregular-shaped flat rocks) to gravel [redacted] The top row of Fig. 6 displays 8 out of a total of 45 images extracted from a video. Each image is processed through the DINO VFM, yielding 1200 patches of dimension $\xi_f = 384$ per image (computed using (42)). 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 3^{rd} and 4^{th} 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 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



(a) (b)

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

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	$\mathbf{Q}$ diag	λ
$\mathbf{\Phi}$ = ct.	0.05	0.1	0.01	0.1	1.0	0.01
$\mathbf{\Phi}$ = NN	0.05	0.1	0.01	0.1	1.0	0.01

simulator were modeled via (11), and the controller of (18) and (19) 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 nominal control matrix $\mathbf{B}_{\text{n}} \in \mathbb{R}^{2\times 2}$ in (11) such that $\mathbf{B}_{\text{n}}$ is replaced by $\eta\mathbf{B}_{\text{n}}$ and the dynamics matrix $\mathbf{A}_{\text{n}} \in \mathbb{R}^{2\times 2}$ is kept the same as in (11).

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 terrains from Fig. 7a. As explained in Sec. V, this model processes high-resolution terrain images into a more compact, lower resolution embedding. This reduced resolution representation is overlaid 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 (42), for a background image of size 480×640, a patch size of $\xi = 16$, and the feature size $\xi_f = 384$.

$$\mathcal{I}_{\text{VFM}} : \left\lfloor \frac{480}{16} \right\rfloor \times \left\lfloor \frac{640}{16} \right\rfloor \times 384 = 30 \times 40 \times 384. \tag{43}$$

Then, we tile each of the DINO feature images across the entire map vertically 4 times $\left(\left\lfloor \frac{120}{30} \right\rfloor\right)$ and horizontally 6 times $\left(\left\lfloor \frac{240}{40} \right\rfloor\right)$ 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 (11) 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 $\mathbf{y}$, computed as in (20). Using this dataset, we then train the basis function $\mathbf{\Phi}$, whose architecture can be seen in Fig. 3, using Algorithm 1. This compact representation of the terrain is then integrated with online adaptive control (Algorithm 2).

B. Simulation Study Objectives

In exploring the capabilities of our model, we investigate how prior knowledge of the terrain contributes 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. 8). b) We test the algorithm under simulated nighttime conditions to gauge performance when the ground is identical, but the lighting conditions are different (Fig. 9). c) We challenge the model by presenting it with two environments that have similar visual features to those in the training data 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. 11).

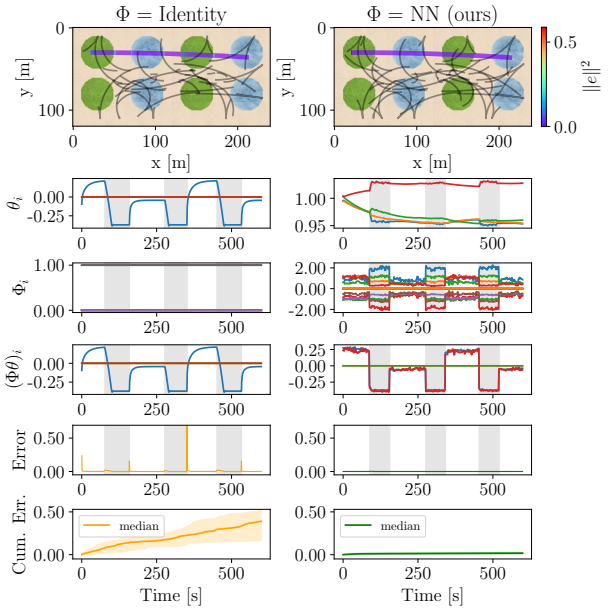


Fig. 8: Results for in-distribution data from the simulation model. Each experiment was run 40 times on the terrains from row 1. The left column contains the performance for the baseline controller, while the right column contains the performance for our method. The second row contains the adaptation coefficients $\hat{\theta}_i$, while the third row emphasizes the basis function $\mathbf{\Phi}_i$. For the baseline, $\mathbf{\Phi}_i$ is constant, while in our method, $\mathbf{\Phi}_i$ varies as a function of the terrain and state. The last row presents the cumulative error, where the thick colored line represents the median, and the shaded region encompasses the range from the 25$^{\text{th}}$ to the 75$^{\text{th}}$ percentile.

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. 8 shows 39 random trials (black) and the single exemplar path (shades of 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 set as constant matrices (44),

$$\mathbf{\Phi} = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right\}. \tag{44}$$

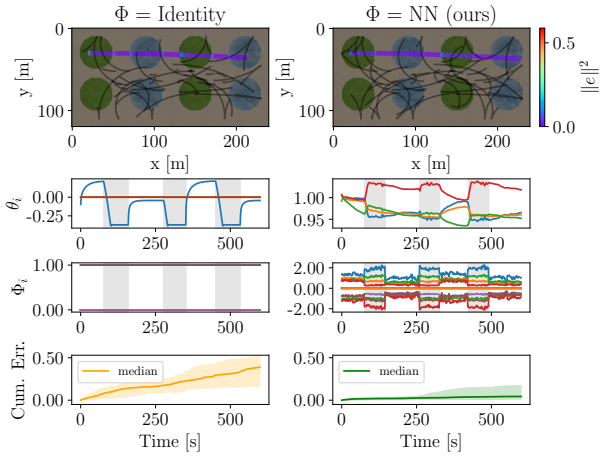


Fig. 9: Simulation results in the simulated nighttime environment. Despite different lighting conditions, the cumulative error is kept small by the terrain-informed DNN.

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 remain relatively stable, while the DNN output itself varies with each terrain type, as shown in the third row of Fig. 8. The fourth row showcases the components of the product between the DNN basis function $\mathbf{\Phi}$ and the adaptation vector $\hat{\boldsymbol{\theta}}$. Though the output of our controller is slightly noisier, the adaptation of the product $\mathbf{\Phi}\hat{\boldsymbol{\theta}}$ 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 negligible, 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 the 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 $\mathbf{\Phi}$, defined as in (44).

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. 9). 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 VFM, underscoring its ability to adapt effectively to varying lighting conditions. Importantly, even in these altered night conditions, the cumulative error remains low.

We highlight the importance of adaptation by comparing the tracking error under two scenarios: with adaptation and without adaptation, as shown in Fig. 10. 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.

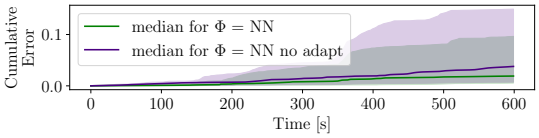


Fig. 10: Comparison of the cumulative error between adaptation and "no adaptation" for the simulated nighttime experiment. In both cases, the DNN version of $\mathbf{\Phi}$ is used. This figure emphasizes the benefit of doing online adaptation.

E. Adversarial Environment Performance

In our final test (Fig. 11), 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.

VII. Empirical Results: Hardware Experiments

We focus on the hardware implementation and experimental validation of our MAGIC$^{\text{VFM}}$ adaptive controller discussed in Sec. III-V on a tracked vehicle whose dynamics are modeled in (11) and a car with Ackermann steering with the dynamics modeled as in (29). We present how our adaptive controller effectively addresses various perturbations such as terrain changes, severe track degradation, and unknown internal robot dynamics.

A. Robot Hardware and Software Stack

Experiments were carried out using a GVR-Bot [94] and a modified Traxxas X-Maxx, both shown in Fig. 12. Both vehicles are equipped with an NVIDIA Jetson Orin, RealSense D457 cameras (GVR-Bot: two forward facing and one rear facing, Traxxas: single forward facing) and a VectorNav VN100 Inertial Measurement Unit (IMU).

State estimation is provided onboard using OpenVINS [95], which fuses the camera data with an IMU to estimate the platform's position, attitude, and velocity. Our MAGIC$^{\text{VFM}}$ controller, as presented in (18), (19) for the tracked vehicle and

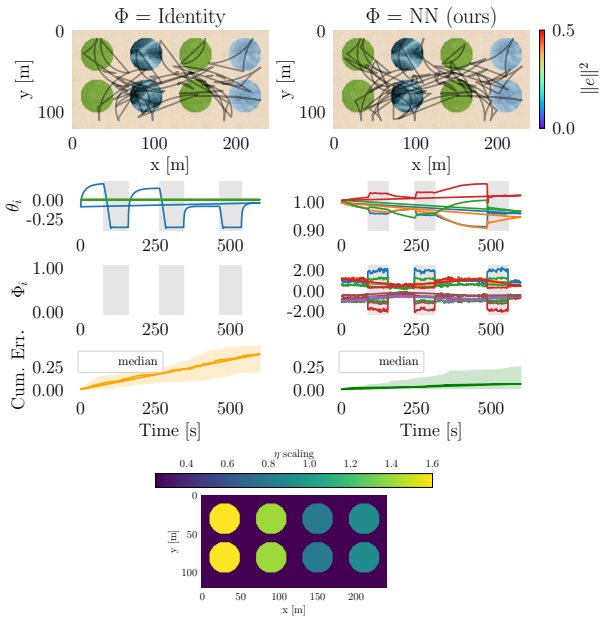


Fig. 11: Simulation results with adversarial environment (different structures that look like ice) and different η scaling coefficient for the control matrix for identical terrain.

in (36) for the car-like vehicle, and Theorem 1, runs at 20 Hz. It is implemented in Python using the Robot Operating System (ROS) as the middleware to communicate with the robot's internal computer.

B. Experiments on Slopes [redacted]

The GVR-Bot only accepts velocity commands as the track velocities are regulated using an internal PID controller, which is inaccessible to the user. While this justifies our first-order modelling (11) using velocities, these experimental results validate that MAGIC$^{\text{VFM}}$ successfully learns the unknown internal dynamics. To verify the performance of our MAGIC$^{\text{VFM}}$ controller (Sec. III and IV) on different terrains, the GVR-Bot was driven on the slopes of the [redacted]. Fig. 12 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 adaptive controller of (18), (19). This function processes inputs comprising of the mean of two visual feature patches from the GVR-Bot's right and left tracks and the robot's velocity taken from the onboard state estimator. The VFM-based DNN ($\mathbf{\Phi}$) 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 $n = 2$ is the state size, $m = 2$ is the control input size, and $n_\theta = 4$ is the size of the adaptation vector that matches the number of terms in the control matrix. The hyperparameters for the training algorithm are shown in Table II.

TABLE II: Training hyperparameters for Algorithm 1 for the tracked vehicle. β is the learning rate for the optimization in Line 11 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

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

k_{px}	k_{py}	k_{dx}	$k_{d\omega}$	k_ψ	$\mathbf{\Gamma}_0$ 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

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 Sec. 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 $\mathbf{\Phi}$. This function, together with an online-adapting vector, is then employed to dynamically adjust the residual $\mathbf{B}$ matrix (18), (19) in real time to account for the different terrains.

The benefits of the terrain-informed basis function can been seen by comparing the performance of a constant and non-constant basis function controller as the robot traverses slopes. Both controllers are based on (18) and the adaptation law in (19). The first controller uses a constant basis function, defined in (44). We choose this structure for the constant $\mathbf{\Phi}$ to capture both the direct and cross-term effects on the robot's velocity. The second controller uses a terrain-dependent DNN basis function trained as explained in Sec. VII-B1. 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. 13. For repeatability, we used a rake to re-distribute the gravel on the slopes between runs and alternated back and forth between running the two controllers. For this experiment, the desired trajectory is a straight line that spans the entire length of the two slopes (see Fig. 12).

Fig. 13 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 Error (RMSE) 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

[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 to maintain the integrity of the VFM features.

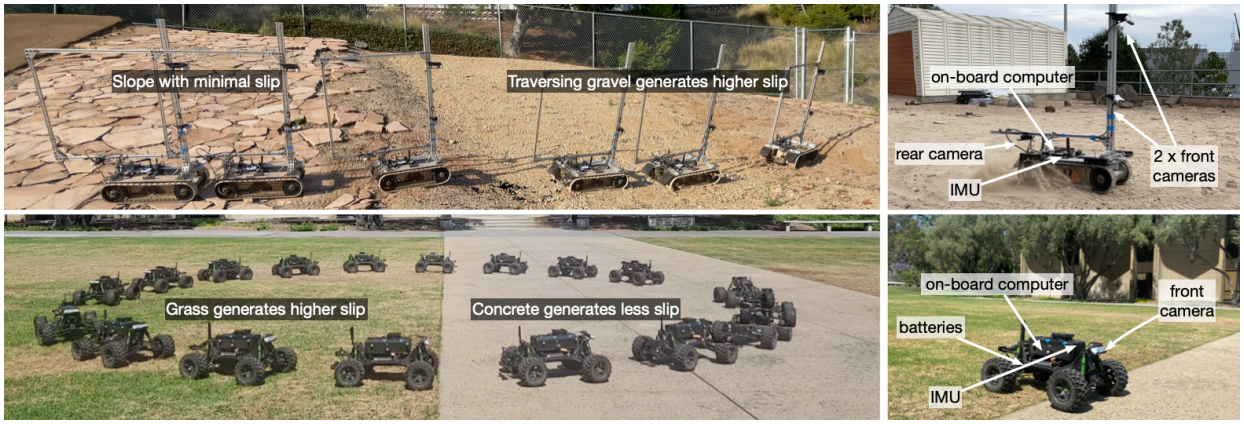


Fig. 12: a) The GVR-Bot traversing two slopes with different textures and terrain-induced dynamic behaviors [redacted]. b) The GVR-Bot with the sensing and compute units highlighted. Note that the forward facing camera is used for state estimation, while the top camera is used for taking terrain images for MAGIC$^{\text{VFM}}$. The rear camera is not used in this work. c) The Traxxas robot traversing two different terrains that induce different dynamic behaviors. d) The Traxxas robot with its main sensing and compute units highlighted.

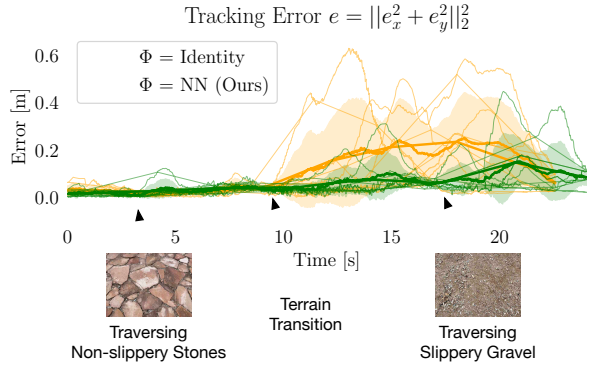


Fig. 13: Tracking error for the two controllers (constant basis function and terrain-dependent basis function) on the slopes for a tracked vehicle. The error is computed as the Euclidean distance between actual and desired positions in the $\mathcal{I}$ frame. For both colors, the thick line outlines the mean of the 5 experiments, the shaded area represents 1 standard deviation, and the thin and transparent lines denote the 5 experiments.

TABLE IV: Statistics for the tracked vehicle on Mars Yard slopes.

Controller	Tracking error (RMS [m])
$\mathbf{\Phi}$ constant	0.130 ± 0.038
$\mathbf{\Phi}$ DNN (ours)	0.061 ± 0.022

enhances tracking performance, yielding an average improvement of 53%.

3) Computational Load: A significant bottleneck in deploying VFMs 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 21 million network parameters. This architecture allows us to run the controller at 20 Hz on the Graphics Processing Unit (GPU) on-board an NVIDIA Jetson Orin.

C. Experiments On-board an Ackermann Steering Vehicle

We performed similar experiments to those described in Sec. VII-B using an Ackermann steering vehicle. Here, the robot traverses two different terrains, as seen in Fig. 12, which induce different dynamic behaviors onto the robot (grass is more slippery than concrete). Our experiments on both vehicles showed that, on flat ground, the car experiences more significant slippage and terrain disturbances compared to the tracked vehicle. Therefore, for this experiment, we validated MAGIC$^{\text{VFM}}$ on flat ground.

TABLE V: Control coefficients for the lateral control of the Ackermann steering vehicle.

Controller	k_p	k_d	$\mathbf{\Gamma}$ diag.	$\mathbf{Q}$ diag.	$\mathbf{R}$ diag.	λ
(a) nonlinear PD	1.0	1.0	-	-	-	-
(b) $\mathbf{\Phi}$ constant	1.0	1.0	1.5	1.0	0.01	0.05
(c) $\mathbf{\Phi}$ DNN	1.0	1.0	1.5	1.0	0.01	0.05

In Fig. 14, we show the product $\mathbf{\Phi}\hat{\boldsymbol{\theta}}$ for the constant basis function of the nonlinear tracking controller in (36). As the robot transitions between the two terrains, we see that the robot effectively adapts to each terrain during this transition. This behavior mirrors that observed in the simulation plots (Fig. 8). Note that we maintained $n_\theta = 4$, to be consistent with the DNN model of the basis function, even though all four parameters are identical in this instance. In Fig. 15, we emphasize the adaptation coefficients (left) and the DNN basis function output (right) for the nonlinear tracking controller in (36) as the robot transitions between the two terrains (grass and concrete) several times. The DNN basis function switches depending on the type of environment it operates in, while the corresponding adaptation coefficients $\hat{\boldsymbol{\theta}}$ remain mostly constant. This behavior also mirrors that observed in the simulation plots (Fig. 8) when a DNN with VFM is employed. Lastly, in Fig. 16, we present the lateral position error ($e^{\perp}$) and lateral velocity ($v_y^{\mathcal{B}}$) for the 3 controllers (a) nonlinear PD ((36) without the adaptation), (b) MAGIC with constant $\mathbf{\Phi}$ in (36) and(19), (c) MAGIC$^{\text{VFM}}$ with DNN $\mathbf{\Phi}$ in (36) and (19). Our method shows superior performance compared to the baseline nonlinear PD controller. The control coefficients for the three controllers are outlined in Table V.

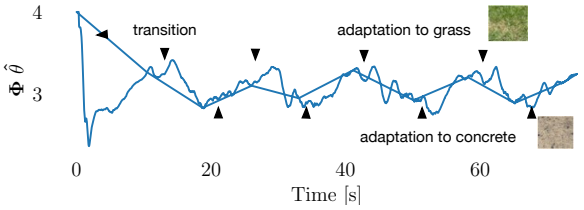


Fig. 14: $\mathbf{\Phi}\hat{\boldsymbol{\theta}}$ of (36) for the constant basis function baseline for the car-like robot. Different terrains induce convergence to different adaptation coefficients.

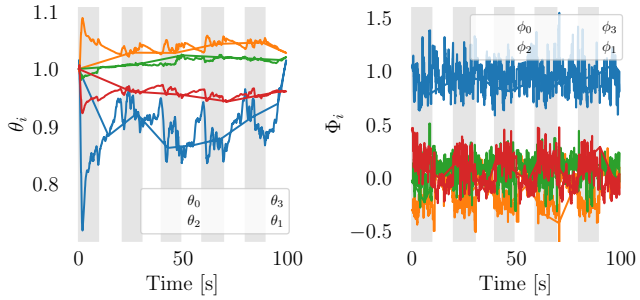


Fig. 15: Adaptation coefficients $\hat{\boldsymbol{\theta}}$ and terrain-dependent DNN basis function for a car-like robot. Approximate under-vehicle terrain is denoted with the white (concrete) and gray (grass) bars.

D. Indoor Track Degradation Experiments

[redacted]

The primary objective of these experiments was to evaluate the robustness and performance of our proposed controller under artificially-induced track degradation. Specifically, the experiments quantify the extent of degradation that our controller can effectively manage and demonstrate its advantage over baseline controllers in similar scenarios. We compare three controllers: (a) nonlinear PD ((18) without the adaptation), (b) MAGIC with constant $\mathbf{\Phi}$ in (18) and (19), and (c) MAGIC$^{\text{VFM}}$ with DNN $\mathbf{\Phi}$ in (18) and (19). The DNN is not retrained on the new ground type, but the previously trained DNN from Sec. 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 RMSE 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 23% and 31%, proving the robustness to model mismatch of both DNN and non-DNN controllers.

E. Performance at DARPA's Learning Introspection Control

The DARPA LINC program [96] develops machine learning-based introspection technologies that enable systems to respond to changes not predicted at design time. LINC took place throughout 2023 at Sandia National Laboratories.

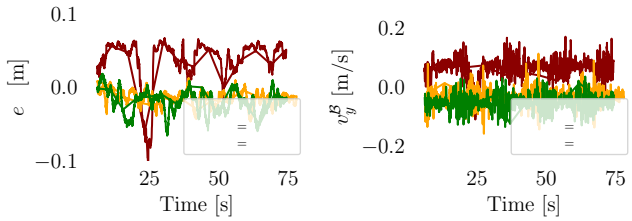


Fig. 16: Convergence of the lateral error $e^{\perp}$ and lateral velocity $v_y^{\mathcal{B}}$ for a circular trajectory traversing two terrains like the one seen in Fig. 12 with $v_x^{\mathcal{B}} = 1.5\ m/s$. The velocity of the desired trajectory is limited by the performance of the Visual Inertial Odometry (VIO) at higher speeds.

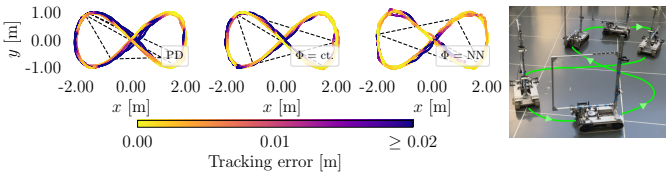


Fig. 17: a) Tracking error for the three controllers during track degradation. (left) the performance for the nonlinear PD (the baseline). (middle and right) performance for the constant basis function and the terrain-informed basis function. b) The fi ure 8s tra ectories for evaluating track degradation erformance [redacted]. The consistent floor [redacted] ensures any slippage is consistent both within the figure 8 and between tests.

The main exercise, Combined Circuit (Fig. 18), evaluated conditions such as track degradation, collisions, tip-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 in order to introduce additional challenges such as adversarial driving and driver intent inference.

For this exercise, we implemented the MAGIC controller from (18), (19) in which the basis function (44) was constant. Trajectories (both position and velocity) were generated using a sampling-based motion planner based on Monte Carlo Tree Search (MCTS), with the desired goal locations generated using a 'driver intent' module that generated a desired path based on operator joystick inputs. 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 and angular velocities) from the VIO with the reference trajectory $\mathbf{v}_{\text{ref}}$ computed from the desired trajectories generated by the MCTS planner, as explained in Sec. IV-C. For the baseline, we compare the desired command from the joystick with the actual state from the VIO.

The following subsections discuss each of the components of the Combined Circuit and the performance of our controller. In Table VII, we present the performance metrics for the four exercises of the LINC project. Each exercise was traversed 4 times and the RMSE of the linear and angular velocity was computed.

1) Chicane Track: The Chicane Track highlighted the rejection of artificially induced track degradation, which was

TABLE VI: Position tracking error statistics for the tracked vehicle experiencing track degradation.

Controller	Tracking RMSE [m]	Improvement
(a) nonlinear PD	0.102	-
(b) $\mathbf{\Phi}$ = constant	0.079	23%
(c) $\mathbf{\Phi}$ = DNN	0.070	31%

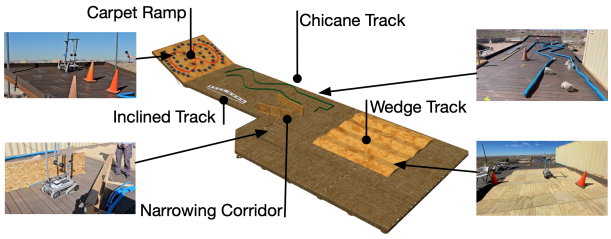


Fig. 18: Combined Circuit for the DARPA Learning Introspective Control (LINC) runs showing the full course with break outs of each of the elements. Track credit: Sandia National Laboratories team.

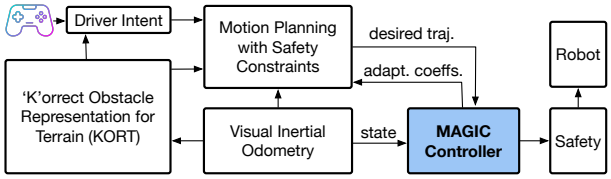


Fig. 19: Architecture for our DARPA LINC software stack, with the MAGIC controller showcased in blue.

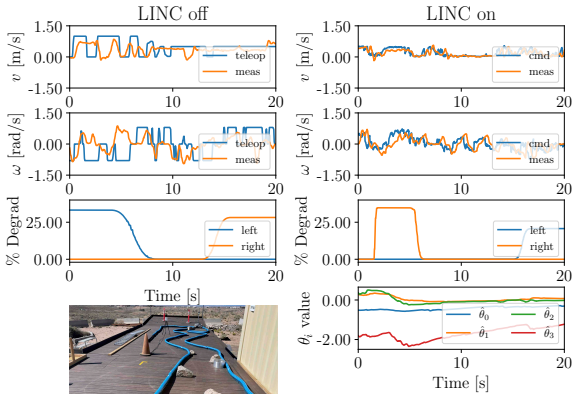


Fig. 20: (Chicane Track) The left and right columns display tracking performance without and with our MAGIC controller, respectively.

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 increase in collisions with the chicane walls if not quickly adapted to. Our MAGIC controller was able to successfully 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. Table VII shows the improved performance of the MAGIC controller on this exercise. Our controller improved linear velocity tracking by 42%, and angular velocity tracking by 19%. Because the track degradation information, $\mathbf{B}_n + \mathbf{\Phi}\hat{\boldsymbol{\theta}}$ of (18), is estimated by the MAGIC controller in real-time, the MCTS can successfully generate trajectories that use this corrected control matrix, thereby successfully avoiding 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 safety.

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 had 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 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, as seen in Table VII.

3) Narrowing Corridor: The aim of the Narrowing Corridor mirrored that of the Chicane Track, assesing the robot's ability to consistently navigate through a tight corridor despite track degradation. For the robot's performance, see Table VII.

4) Wedge Track: Similar to the Carpet Ramp exercise in Sec. VII-E2, the Wedge Track tests the ability of the algorithms to maintain control and slow down under slippage while minimizing the risk of tipping. When traversing discrete wooden wedges, the robot often loses traction. Moreover, the robot experiences sudden positive and negative accelerations due to the downhill and uphill traversal of a wedge pair. 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 velocity spikes as it traverses the wedges. These rapid changes are caused by the VIO's Kalman filter that integrates spikes measured by the accelerometer when the robot bounces off the wedges. The bottom row of Fig. 22 shows the adaptation coefficients quickly adapting for the loss of traction.

5) MAGIC and Human-in-the-Loop: The LINC program was different from many robotics projects in that the global planner was human-driven rather than autonomous. This

TABLE VII: Performance metrics for the four exercises of the DARPA 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	0.45	16.36	0.96	0.59	19.96	0.37	0.58	34.95	0.52	0.72	17.95
LINC on	0.16	0.36	44.36	0.36	0.32	39.22	0.25	0.34	39.72	0.19	0.44	23.73
Improvement	**42%**	**19%**		**62%**	**46%**		**33%**	**41%**		**64%**	**39%**	

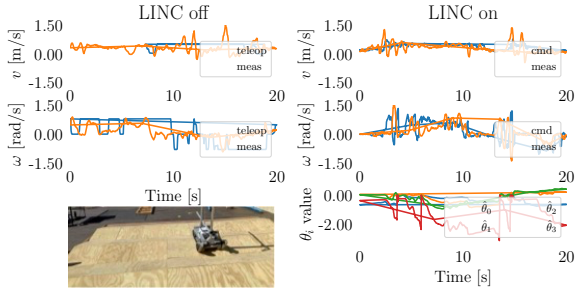


Fig. 21: (Carpet Ramp) The left and right columns display tracking performance without and with our MAGIC controller.

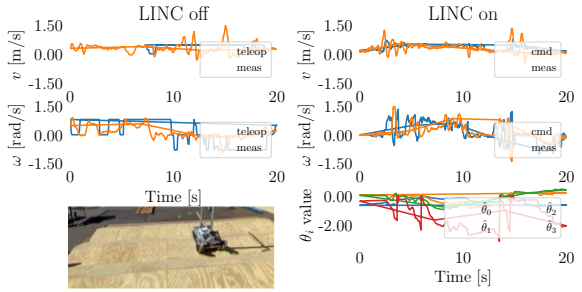


Fig. 22: Trajectory tracking during the Wedge Track run. The right and left columns display tracking performance without and with our controller. No artificial degradation was introduced by the organizers.

presents a challenge as MAGIC must not degrade the user driving experience but instead must augment it without the forward-planning and control input smoothness assumptions of typical robotic projects. 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, and secondly, much of the adaptation to the changing terrain and vehicle were significantly reduced by MAGIC.

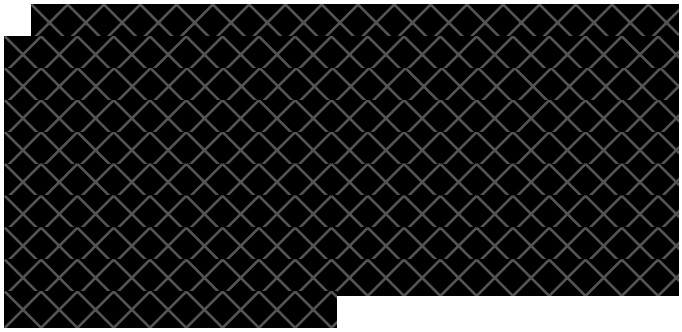
VIII. Conclusion

We 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% decrease in position tracking error when deployed on a tracked vehicle traversing two different sloped terrains. We further demonstrated our algorithm onboard a car-like vehicle and showed that the learnt DNN basis function captures the residual dynamics generated by the two different 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 as part of the DARPA's LINC project.

Acknowledgement



References

[1] M. M. Foglia and G. Reina, "Agricultural robot for radicchio harvesting," *J. 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 Robot. Autom. Letters*, vol. 2, pp. 664–671, Apr. 2017.

[5] Z. Kashino, G. Nejat, and B. Benhabib, "Aerial wilderness search and rescue with ground support," *J. Intelligent Robotic Syst.*, 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. Veh. Technol.*, 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," *J. 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," *J. 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. 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: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. Intelligent Robots 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. 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. Pattern Anal. Mach. Intell.*, 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. 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. 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, 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," *J. 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 Robot. Autom. 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. Autom. 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. Intell. Robot. Syst.*, 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. Intell. Robot. Sys. (IROS)*, pp. 7661–7666, Oct. 2020.

[32] X. Song *et al.*, "Rapidly adaptable legged robots via evolutionary meta-learning," *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Oct. 2020.

[33] S. Belkhale *et al.*, "Model-based meta-reinforcement learning for flight with suspended payloads," *IEEE Robot. Autom. 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 Robot. Autom. Letters*, vol. 6, no. 2, pp. 3240–3247, 2021.

[35] B. Rothrock *et al.*, "SPOC: Deep learning-based terrain classification for Mars Rover missions," in *AIAA SPACE 2016*, Amer. Inst. Aeronaut. 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] G. Shi, K. Azizzadenesheli, M. O'Connell, S.-J. Chung, and Y. Yue, "Meta-adaptive nonlinear control: Theory and algorithms," *Advances Neural Information Processing Systems*, 2021.

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

[39] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," in *Proc. 36th International Conference on Machine Learning*, pp. 1920–1930, June 2019.

[40] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel, "Continuous adaptation via meta-learning in nonstationary and competitive environments," *arXiv:1710.03641*, 2017.

[41] 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.

[42] J. Harrison, A. Sharma, C. Finn, and M. Pavone, "Continuous meta-learning without tasks," *arXiv:1912.08866*, 2019.

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

[44] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Advances in Neural Information Processing Systems*, pp. 2451–2463, 2018.

[45] 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, 2022.

[46] 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.

[47] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," *arXiv:2309.14341*, 2023.

[48] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[49] P. R. Florence, L. Manuelli, and R. Tedrake, "Self-supervised correspondence in visuomotor policy learning," *IEEE Robot. Autom. Letters*, vol. 5, pp. 492–499, 2019.

[50] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *arXiv:1812.00568*, 2018.

[51] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 5026–5033, Oct. 2012.

[52] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robot. Autom. 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 Robot. Autom. 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," in *Proc. Robot.: Sci. Syst.*, 2023.

[55] M. Oquab *et al.*, "DINOv2: Learning robust visual features without supervision," *arXiv:2304.07193*, 2023.

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

[57] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv: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. Learning Representations*, 2022.

[59] G. Erni, J. Frey, T. Miki, M. Mattamala, and M. Hutter, "Mem: Multi-modal elevation mapping for robotics and learning," in *2023 IEEE/RSJ International Conf. Intell. Robot. Syst. (IROS)*, pp. 11011–11018, 2023.

[60] Y. Ze *et al.*, "GNFactor: Multi-task real robot learning with generalizable neural feature fields," *arXiv:2308.16891*, 2023.

[61] Y. Wang *et al.*, "D^3fields: Dynamic 3D descriptor fields for zero-shot generalizable robotic manipulation," *arXiv:2309.16118*, 2023.

[62] M. Naseer *et al.*, "Intriguing properties of vision transformers," in *Advances in Neural Information Processing Syst.*, 2021.

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

[64] 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. Robot. Autom.*, pp. 5321–5327, May 2020.

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

[66] J.-J. E. Slotine and W. Li, "Composite adaptive control of robot manipulators," *Automatica*, vol. 25, no. 4, pp. 509–519, 1989.

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

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

[69] M. Sombolestan, Y. Chen, and Q. Nguyen, "Adaptive force-based control for legged robots," in *2021 IEEE/RSJ Intern. Conf. Intell. Robot. Syst. (IROS)*, pp. 7440–7447, Sept. 2021.

[70] K. Tsujita, K. Tsuchiya, and A. Onat, "Adaptive gait pattern control of a quadruped locomotion robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, vol. 4, pp. 2318–2325 vol.4, 2001.

[71] P. Petrov and F. Nashashibi, "Modeling and nonlinear adaptive control for autonomous vehicle overtaking," *IEEE Trans. Intell. Transport. Syst.*, vol. 15, no. 4, pp. 1643–1656, 2014.

[72] 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.

[73] 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.

[74] J. Seo, T. Kim, S. Ahn, and K. Kwak, "Metaverse: Meta-learning traversability cost map for off-road navigation," *arXiv:2307.13991*, July 2023.

[75] 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. Intell. Robot. Syst.*, 2023.

[76] S. Banerjee, J. Harrison, P. M. Furlong, and M. Pavone, "Adaptive meta-learning for identification of rover-terrain dynamics," *arXiv:2009.10191*, 2020.

[77] 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.

[78] 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 Control*, vol. 52, pp. 135–169, Oct. 2021.

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

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

[81] 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.

[82] 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.

[83] M. Mistry and L. Righetti, "Operational space control of constrained and underactuated systems," in *Proc. Robotics: Science Syst.*, June 2011.

[84] 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. Robot.*, vol. 21, pp. 834–849, Oct. 2005.

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

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

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

[88] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for non-linear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

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

[90] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *Amer. Control Conference*, pp. 2296–2301, July 2007.

[91] M. Caron *et al.*, "Emerging properties in self-supervised vision transformers," *arXiv:2104.14294*, 2021.

[92] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. and their Applications*, vol. 13, . 18–28, Jul 1998.

[93] [redacted]

[94] US Army CCDC GVSC, "GVR-BOT Users Guide," 2019.

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

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