Dynamics Modeling using Visual Terrain Features for High-Speed Autonomous Off-Road Driving

Author Names Omitted for Anonymous Review. Paper-ID 20

***Abstract*—Rapid autonomous traversal of unstructured terrain is essential for scenarios such as disaster response, search and rescue, and planetary exploration. As a vehicle navigates at the limit of its capabilities over extreme terrain, its dynamics can change suddenly and dramatically. For example, varying terrain can affect parameters such as traction, tire slip, and rolling resistance. To achieve effective planning in such environments, it is crucial to have a dynamics model that can accurately anticipate these conditions and respond before an issue can occur. In this work, we present a hybrid model that predicts the changing dynamics induced by the terrain as a function of visual inputs. We leverage a pre-trained visual foundation model (VFM) DINOv2, which provides rich features that encodes fine-grained semantic information. To use this dynamics model for planning, we propose an end-to-end training architecture for a projection distance independent feature encoder that compresses the information from the VFM, enabling the creation of a lightweight map of the environment at runtime. We validate our architecture on an extensive dataset (hundreds of kilometers of aggressive off-road driving) collected across multiple locations. (Video link omitted for double-blind review.)**

I. Introduction

Dynamics modeling for field robots using visual inputs has been an area of research since before visual foundation models (VFMs) were available. For example, in [1], different vehicle slip models were determined for a small number of different terrain conditions (such as grass, gravel, etc.). Though discretizing terrain types could suffice in some environments and when the robot is not operating at its hardware limits, a continuous representation of terrain is more desirable for our purposes, where terrain varies drastically and the driving is highly aggressive.

VFMs are large-scale, pre-trained neural networks designed for diverse vision tasks like classification, segmentation, and feature extraction. Models like DINOv2 [14] generate general-purpose feature vectors that provide a good basis for zero-shot adaptation. In this work, we leverage DINOv2 to extract compact, continuous terrain representations that capture visual details relevant to vehicle dynamics.

Understanding the vehicle's surrounding terrain is especially important as the coupling of high speeds and varying ground terrain introduce complex, nonlinear, and time-varying properties in the dynamics of the vehicle as aspects such as the traction, cornering stiffness, and rolling resistance of the vehicle change. For example, the vehicle's dynamics when driving on slippery grass differ from those on dry trails. Examples of terrain where the vehicles dynamics may be affected as a result of the terradynamics of the environment are shown in Fig. 2.

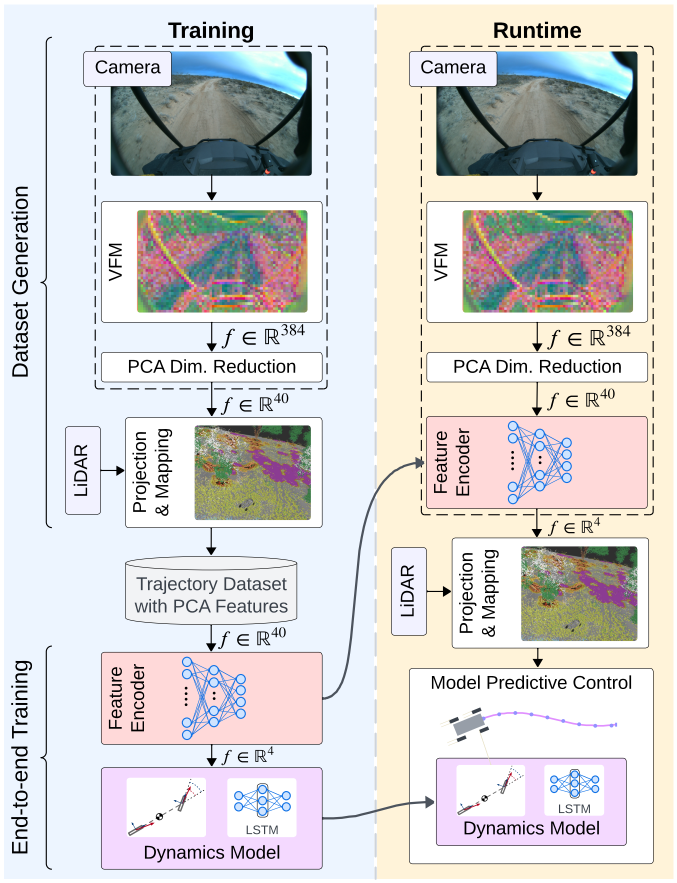


Fig. 1. Architecture of dynamics learning with visual features. A feature encoder is trained end-to-end with the dynamics model on a trajectory dataset including high-dimensional visual features from a VFM. This encoder reduces the visual information to a low-dimensional, dynamics relevant feature space. At runtime, it processes features in image space (dashed box) before projection and accumulation in a 3D map. This makes the map aggregation step computationally tractable. The 3D map is flattened to a top down 2D map used by the dynamics model in the MPC planner.

These challenges highlight the importance of integrating perceptual inputs into our dynamics model. Most importantly, the dependence on visual features will allow for the *anticipation* of changes to vehicle dynamics, ultimately reducing the need for model correction and adaptation as the vehicle drives on varying terrain. To this end, the contributions of this work are as follows: **1).** A hybrid dynamics model (i.e., a model with both physics-based parametric and neural-network components) that predicts changes in terradynamics as a function of terrain using visual features derived from

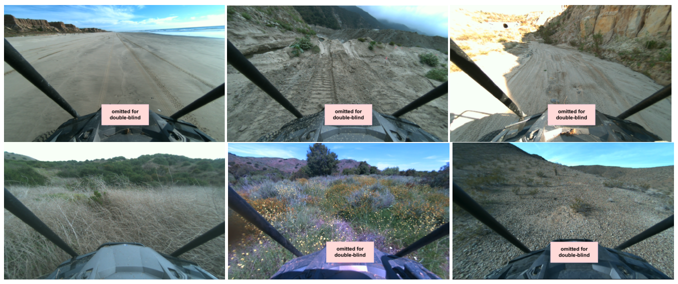


Fig. 2. Terrain geometries and properties vary significantly across the environments. Images show a selection of diverse terrain (from top left to bottom right: packed sand, muddy ditches and ruts, loose dirt trail, tall grass, dense overgrown vegetation, steep slopes) for which visual inputs of the terrain inform the changing dynamics of the vehicle.

a VFM. **2).** A training architecture that enables end-to-end learning of a feature encoder that can be used to reduce the computational burden of tracking high-dimensional features in the map-space at runtime. **3).** A novel method for compressing a visual feature space that is robust to projection distance and occlusion, enabling generalization to real driving scenarios. **4).** Validation of our method on a large dataset of aggressive off-road driving across four different rugged terrains.

II. Related Work

In off-road autonomy literature, many studies aim to improve the quality of the traversability map by incorporating visual information about the surrounding terrain [16], [2], [5], [6], [13], [7]. For example, [16] combines semantic labels and geometric hazard identification to determine the costs on the traversability map. [2] learns a traction model to estimate the slip parameters of the terrain and factors this into the cost of the traversability map. Some works, such as [7], classify terrain into a discrete set of options, such as a smooth and rough regolith, limiting their approach. The main goal of these methods is to avoid unsafe paths. Less explored is the use of visual terrain features to inform vehicle dynamics modeling, which is necessary for accurate control and planning at a higher resolution.

Other works, such as [11] and [9], also propose hybrid models (i.e., a model that has parametric and learned components) for high-speed vehicle dynamics modeling. In both works, the vehicle drives on-road, and therefore neither approach uses visual terrain features to inform the model.

Most relevant to this research are [12] and [3]. Both methods use the DINO or DINOv2 VFM to inform a model about the traversed terrain. [3] employs the visual features to inform two physical parameters, stiffness and friction, while [12] uses this visual terrain-feature-based model for adaptive control. Compared to [3], our method outputs a richer representation of the terrain by training a feature encoder end-to-end with a dynamics model. In contrast to [12], which processes the terrain information at each camera frame and uses that instantaneous information for control, our method incorporates a lower-dimensional representation of the DINOv2 features into a 2D map, which is then queried at each of the wheel locations. This allows our method to accurately capture the spatial distribution of terrain features and terrain transitions, which is particularly relevant for predicting trajectories in a Model Predictive Control (MPC) framework.

Perception challenges such as lighting conditions, distance bias, and occlusions can all contribute to the lack of complete and immutable perception of surrounding terrain. Several works address the issue of consistency of data from the vehicle's surroundings during driving. For instance, [8] proposes a self-supervised learning framework that uses both a VFM and human driving trajectories for terrain traversability learning. Similarly, [19] uses human driving data and weakly-supervised contrastive learning. Labels of the surrounding terrain may change as the vehicle drives, and previously unseen voxels may be filled in as the vehicle passes through occlusions such as bushes. Multimodal mapping approaches like [4] have used accumulation strategies such as latest information, exponential averaging and Bayesian updates. Pyramid occupancy networks have also been used to accumulate map predictions across timesteps. For example, [15] uses Bayesian filtering to fuse voxel map information over a time buffer of driving. To further analyze these perception challenges, we examine the consistency of the DINOv2 features with respect to the distance at which they are collected and the dimensionality reduction we perform.

III. Mapping of Visual Features for Controls

To provide context, we briefly introduce the main components of the autonomy stack used in this work. A mapping module [omitted for double-blind review] combines image and LiDAR information by projecting the pointclouds onto image data. This module then aggregates the resulting pointclouds, augmented with the image data, in a 3D voxel map. Depending on the downstream application, the image data can be represented as semantic class probabilities or as a latent feature space. This 3D voxel map is further processed into a 2D traversability map, which is used by the planning and control modules. This 2D traversability map contains multiple layers that encode various quantities such as elevation, obstacles, and planning costs. It also contains terrain features for the dynamics model in this work.

A. Model Predictive Control with Learned Dynamics

As in our previous work [omitted for double-blind review], we learn a dynamics model for Model Predictive Control (MPC), to plan optimal trajectories for the vehicle. We use Model Predictive Path Integral (MPPI) [18], a sampling-based MPC method. It operates by sampling various trajectories, performing forward rollouts of the dynamics, and optimizing the trajectories based on the cost of these sampled rollouts. MPPI is well-suited for our application since it is easily parallelizable on the GPU [17] and supports the use of complex, sparse cost functions. The costs are made up of soft penalties to encourage good behavior and discontinuous penalty functions for safety critical constraints. The specific cost function and the MPPI variation we use are described in [omitted for double-blind

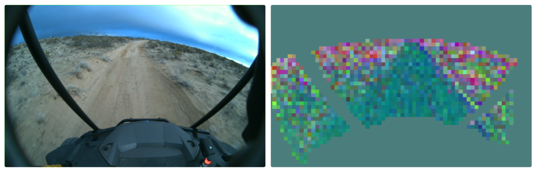


Fig. 3. Left: A forward-facing image of size $\mathbb{R}^{960\times594\times3}$ (in RGB). Right: VFM output of size $\mathbb{R}^{68\times42\times384}$, where each 14×14 pixel patch results in one feature vector of size $\mathbb{R}^{1\times384}$. DINOv2 features from ground regions undergo PCA, and the first three components are visualized in RGB. The result effectively segments on- and off-trail terrain.

review]. The trajectory is optimized over a 5-second prediction horizon, requiring accurate and computationally efficient dynamics modeling for effective planning. The terrain elevation and visual terrain features in the 2D traversability map are queried by the dynamics model at the location of each wheel along the trajectory rollout.

B. Dataset Collection Pipeline

We generate a dynamics dataset with visual features by processing recorded driving data with the autonomy stack. The perception stack, which is nearly identical to the one used at runtime, is fed with recorded sensor data and relevant outputs are recorded. The key difference between the runtime and replay configuration is the placement of the feature encoder, as emphasized in Fig. 1. The feature encoder, described further in Section V, compresses visual features into a low-dimensional, dynamics-relevant feature space. When creating the dataset, we store a high-dimensional feature vector to train the feature encoder jointly with the dynamics model. At runtime, the feature encoder runs before projection and mapping, reducing the computation and memory burden of mapping visual features to enable real-time execution.

1) Visual Features: As the vehicle drives, images are captured from four RGB cameras facing forward, back, and to both sides. The front and side cameras operate at a rate of 10Hz, while the rear camera captures images at 2Hz, all with a resolution of 960×594 pixels. The camera images are rectified before being processed by the DINOv2 VFM. We employ the smallest distilled ViT-S/14 network size which uses an embedding dimension of $\mathbb{R}^{384}$.

2) Dimensionality Reduction using Principal Component Analysis: To lower the memory and computational requirements during the dataset generation, we apply Principal Component Analysis (PCA) projection on the output of the VFM (Fig. 1). To generate the PCA basis, a set of 175 images were manually selected, covering various terrain types and lighting conditions. These images are run through the VFM to obtain feature images, which are then masked to remove pixels that are the sky, and a PCA basis is computed from these. The PCA projection provides a linear map from $\mathbb{R}^{n_\text{vfm}} \mapsto \mathbb{R}^{n_\text{pca}}$. The replay can run slower than real time, so it can support a high feature dimension. The size of n_pca is a trade-off between dataset processing speed/memory requirements and model quality. We choose $n_\text{pca} = 40$, which greatly reduces the memory requirements while maintaining a sufficiently rich feature space. As shown in Section VI-A and Fig. 4c, the performance is not highly sensitive to the PCA basis size. An example image and the corresponding RGB visualization of the first three components of the PCA features are shown in Fig. 3.

3) Mapping of Visual Features: The visual features are mapped into a local, robot-centric map by the perception stack. In the case of offline dataset generation, the features are the output of the PCA compression, whereas at runtime they are the output of the feature encoder, which substantially compresses the features. To map these features, LiDAR pointcloud data is projected onto the image plane of the feature image, and the pointcloud values are augmented with the corresponding feature vector. The pointcloud is then aggregated temporally into a 3D voxel map with 0.2 m resolution. To fuse multiple measurements, we retain the closest observation, allowing for both instant updates and temporal stability when moving away from a location [omitted for double-blind review]. The visual terrain features in the 3D voxel data are compressed to a 2D map output by taking the lowest valid data point within each vertical stack of voxels, under the assumption that the visual feature of the ground, and not anything above, is most relevant for dynamics learning. To close small gaps of missing data due to LiDAR sparsity or small occlusions, missing data is filled in with data from the nearest neighboring cell in a 0.4 m radius.

4) Dataset Extraction: To generate a dataset for dynamics learning, the vehicle state and control trajectories are stored together with the visual features and the elevation surface normals under each wheel. This trajectory is later split up into chunks that form the actual 5 s trajectory prediction dataset. Visual features may vary with distance, so we store multiple values in the map from different points in time. In particular, the back camera and LiDAR typically fill in data behind small vegetation. The maps from which to take features are chosen based on the distance between the query point and the map origin (approximately the latest robot location). As a result, each trajectory, at every point along it, will have multiple sets of features mapped from various distances, which we use later to train a distance-independent encoder. In addition, features from a "hindsight" map, which contains the last valid data (and therefore the least amount of missing data), are stored.

IV. Hybrid Vehicle Dynamics Model

Our previous work on dynamics modeling [omitted for double-blind review], has drawn upon hybrid models, which include both physics-based parametric and neural network components. This approach was chosen primarily for the reliability of the parametric components when the networks are in low data regimes. The vehicle dynamics are divided into four main components: brake, steering, engine, and terradynamics. For the first three modules, we model the delay in actuation or RPM (revolutions per minute) as a state. In this current work, we model each component with a hybrid model, except for the engine dynamics, for which we predict the RPM of the engine directly from the throttle and speed of the vehicle. A key difference from our previous work is that we compute compensations to predicted forces rather than directly predict the state. We predict the state vector $\mathbf{x} \in \mathbb{R}^6$ for the terradynamics model containing the inertial positions $\mathbf{p} = [p_x, p_y]^\top$, the yaw angle ϕ, the velocity in the body frame $\mathbf{v} = [v_x, v_y]^\top$, and the angular rate r. An additional 4 states are predicted, such as pressure for the brake, position and velocity for the steering angle, and a value for the engine RPM, but their models are fixed during training of the terradynamics. All constants and networks are learned using the Adam optimization algorithm [10] on training data drawn from a variety of field test sites.

A. Parametric Bicycle Model

To model the parametric portion of the terradynamics hybrid model, we employ a bicycle dynamics model, largely drawn from [9] in form. We include our own computation of forward force using the predicted engine RPM and commanded throttle. We modify the Pacejka tire model and yaw rate equation to avoid integrator stability issues.

$$\mathbf{F} = \begin{bmatrix} \left(P\left(x_{rpm}\right)P\left(u_{th}\right) - P\left(x_{br}\right) - \beta\left(v_x\right)\right)\eta_z \\ \left(D_R \sin\left(C_R \tanh\left(B_R \alpha_R\right)\right)\right)\eta_z \\ \left(D_F \sin\left(C_F \tanh\left(B_F \alpha_F\right)\right)\right)\eta_z \\ \left(\frac{v_x}{C_L}\delta\right)C_r - C_{r,d}r \end{bmatrix}, \tag{1}$$

where $\mathbf{F} = [F_x, F_{yf}, F_{yb}, F_r]^\top$ defines a vector of forces, x_{rpm}, x_{br}, δ are state variables computed by the delay models (engine, brake, steering), u_{th} is the commanded throttle, η_z is the current normal vector from the elevation map rotated into body frame and averaged over the wheels, and D, B, C etc. are all fit constants. $P(\cdot)$ is a quadratic polynomial of the input and $\beta(\cdot)$ is a scaled $\tanh$ of the input. The first equation in (1) is the forward force applied to each wheel, the second two are the lateral forces on the tires, and the final equation is an approximation for yaw rate. The front and rear wheel slip angles, α_F and α_R,

$$\alpha_R = \arctan\left(\frac{v_y - L_R r}{\max(C_{max}, v_x)}\right), \tag{2}$$

$$\alpha_F = \arctan\left(\frac{v_y + L_F r}{\max(C_{max}, v_x)}\right) - \delta, \tag{3}$$

where C_{max}, L_R, L_F are learned parameters. C_{max} controls the stiffness of the slip angle equation, creating a trade-off between the slip angle accuracy and the stiffness.

Finally, the forces are then converted to body frame using geometric transforms $h(\mathbf{F}, \mathbf{x}) : \mathbb{R}^4 \times \mathbb{R}^6 \to \mathbb{R}^6$ to compute the derivatives of body rates using

$$\begin{aligned} \dot{v}_x &= \frac{(1+\cos\delta)F_x - F_{yf}\sin\delta}{m} - C_{x,d}v_x^2 - C_{x,g}\eta_x + v_y r, \\ \dot{v}_y &= \frac{F_{yb} + \cos\delta F_{yf} + F_x \sin\delta}{m} - C_{y,d}v_y^2 - C_{y,g}\eta_y - v_x r, \\ \dot{r} &= F_r, \\ \dot{\mathbf{p}} &= R(\phi)\mathbf{v}, \quad \dot{\phi} = r, \end{aligned} \tag{4}$$

where $C_{\cdot,d}, C_{\cdot,g}$ are the drag and gravity coefficients, and m is the vehicle mass. The system in (4) is integrated using forward Euler integration with a $\Delta t = 0.02s$.

We predict a compensation of the parametric force $\mathbf{F}$ using the neural networks,

$$\dot{\hat{\mathbf{x}}}_t = h\Big(\hat{\mathbf{F}}_t + \zeta\Big(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{y}_t, \hat{\mathbf{F}}_t\Big), \hat{\mathbf{x}}_t\Big), \tag{5}$$

where $\hat{\mathbf{x}}_t$ is the predicted state at time t, $\mathbf{u}_t, \mathbf{y}_t$ are the control and map inputs respectively, and ζ is an LSTM initialized as in [omitted for double-blind review] over a local horizon of historical values $[t-\tau, t], \tau = 0.2$. We emphasize that $\hat{\mathbf{x}}_t, \hat{\mathbf{F}}_t$ are the predicted values from delay models and dynamics, while $\mathbf{u}_t, \mathbf{y}_t$ are what was actually seen or commanded.

V. Feature-Based Dynamics Model

Not all DINOv2 features of the surrounding terrain will be relevant to the dynamics of the vehicle. For example, any feature maps that encode lighting conditions or depth information should not be used in dynamics modeling. We incorporate a feature encoder $\zeta_E \in \mathbb{R}^{n_{\text{pca}}} \mapsto \mathbb{R}^{n_{\text{encoder}}}$, therefore, to compress the feature space into a more compact, dynamically-relevant subspace and train the network within the training pipeline to ensure that only the feature information correlated with dynamics is retained. Each wheel location is compressed individually giving $\mathbb{R}^{4 \cdot n_{\text{encoder}}}$ features that are used as additional inputs to the neural network. The feature encoder is a fully-connected neural network that processes the VFM features extracted at each tire location, resulting in a 40-dimensional input layer run 4 times independently for each wheel. It includes hidden layers of size $[64, 32]$ with $\tanh$ activation functions, followed by an output layer of size n_{encoder}, which we vary in 4b.

A. Learning Distance-Independent Compression

For the terradynamics model, we follow this baseline structure but add an additional network ζ_E that passes encodings of the DINOv2 terrain features into the input of the LSTM ζ_μ as follows:

$$\dot{\hat{\mathbf{x}}}_t = h\Big(\hat{\mathbf{F}}_t + \zeta_\mu\Big(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{y}_t, \hat{\mathbf{F}}_t, \zeta_E(\hat{\mathbf{y}}_f)\Big), \hat{\mathbf{x}}_t\Big). \tag{6}$$

During operation, the map will include features from various projection distances, requiring our network to effectively function within this distance-varying feature space. In order to train our network for these conditions, we collect features from 7 different distance buckets, each spaced out by 10 m and at a maximum of 40 m in front of the vehicle. The exact projection distance is nominally in the range of ± 5 m but can have a large percentage of outliers due to the nature of the processing pipeline and environment.

The DINOv2 features change significantly based on distance. We have previously discussed the many issues that can cause the features to change, but all of these issues remain at runtime so we briefly discuss the impact they have on the features themselves. We ignore invalid values when computing the mean errors and see substantial l_2 differences in feature vectors as a function of projection distance. Even when looking at buckets that are next to each other in distance, we still see features varying by 30% of the total range of values across the dataset for that specific feature. Furthermore, occlusion causes substantial issues as a function of distance; for distance buckets "hindsight," $-20m$, $-10m$, ..., $40m$, we see the following occlusion percentages: $[1, 0.87, 0.92, 0.92, 0.84, 0.65, 0.42, 0.21]$. A negative distance means we have driven over the location and had the chance to collect data using the back LiDAR. The decrease in validity in the closer distance buckets comes from a new ground plane after the vehicle has compressed the traversed ground combined with the slower frame rate on the back camera, so the voxels lack visual features.

We propose randomizing the projection distance by drawing features from any of the 7 distance buckets or hindsight. A random distance is sampled uniformly and is maintained throughout the trajectory, as varying it at each time step resulted in poor performance. We expect that the temporal dependence of invalid values is important for stable results when constrained to a single bucket of distance. We augment the input features to the compression network with a flag $\{-1, 1\}$ to indicate if they are missing and replace the missing features with the mean of the training dataset.

VI. Results

Our results section is divided into two parts. First, we demonstrate the improvements to the dynamics modeling that terrain-based features can provide when the features are purely derived from hindsight. Second, we show that our approach is able to handle features from farther distances when naive methods fail.

All networks are trained for 30 epochs on a dataset of $\approx 2M$ 5-second trajectories with a test set of size $\approx 250k$ trajectories. All summary statistics are computed at the end of the prediction horizon. Initialization LSTMs have 20 hidden layers, and the predictor LSTM uses a hidden size of 4 with an additional output network that transforms the output dimension with a 20 neuron hidden layer. The training and test sets are derived from the same logs but have no overlap. To include correct delays coming from autonomous operation of the vehicle (brake, engine, steering), data is drawn from autonomous driving logs collected over a year from four distinct environments in three locations. Most data (64%) comes from the Mojave Desert near Helendale, CA, featuring loose sand and compressible creosote bushes (see Fig. 2 top middle & right, bottom right). The second environment (28%) is from Halter Ranch near Paso Robles, CA (Fig. 2 bottom left) and includes dry grasses, oak woodland, and steep slopes up to 40°. The final two environments are coastal dunes (2 top left) (4%) and coastal sage scrub (i.e., thick vegetation 2 bottom middle) (4%) collected near Oceanside, CA. From anecdotal experience, all environments bring about unique vehicle dynamics.

A major challenge across environments is ground-plane estimation and occlusion at different distances. Helendale's low vegetation density allows for easy ground-plane estimation, while Halter Ranch's 1-2 ft tall grass makes that estimation difficult, often leading to invalid DINOv2 features. Furthermore, the trees in Halter Ranch and dense vegetation in Oceanside prevent us from observing the occluded terrain before traversing past it. These issues, while not present when using DINOv2 for short-horizon work (such as in [12]), create room for improvement, but we highlight how our method can handle features collected at various distances.

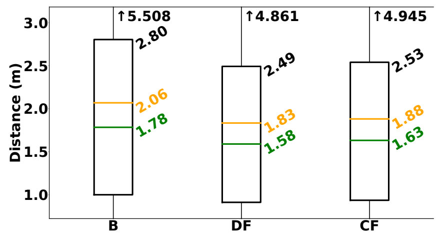
A. Post-Processed Visual Features

On the post-processed visual features, we see that networks with DINOv2 information as inputs outperform the model without any visual information as shown in Fig. 4a. All models perform well since the median distance traveled by a trajectory in the dataset is ≈ 24m with a median speed of ≈ 4.7m/s. The compression scheme (*CF*) performs slightly worse than directly inputting the features into the network (*DF*), likely because of the time-dependence of visual features. Note that naively inputting the (raw) features is not feasible in real time because of the mapping pipeline's sensitivity to additional features. Still there is a $\sim 10\%$ decrease in mean summed loss with the compressed version as compared to the absence of DINOv2 features. Most of the error reduction comes from $\sim 8\%$ reduction in the mean error for v_x at the end of the time horizon $5s$. This directly impacts the accuracy for the yaw since the parametric yaw rate is heavily dependent on speed.

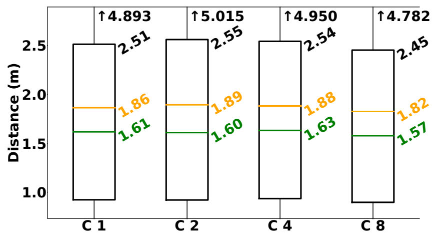
We vary the output dimension n_{encoder} in Fig. 4b and vary the number of PCA features in Fig. 4c to check sensitivity to these parameters. Smaller compression sizes n_{encoder} significantly reduce the memory and computation requirements of the mapping pipeline, so the smallest vector necessary should be used. We see similar performance as we vary these parameters, indicating that compression does not lose useful dynamical information. The PCA feature sensitivity is irrelevant at runtime but can speed up post-processing. Overall, our method is robust to these variations.

B. Varying Distance Features

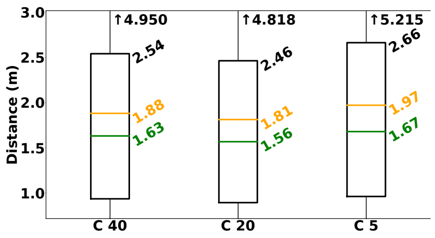
In the previous section, we showed that hindsight features are able to improve the prediction accuracy, but these methods rely on hindsight feature information that does not exist at runtime. We next evaluate our models on features collected at distances more representative of those during vehicle operation, and we see that the networks trained on hindsight features fail in Fig. 5a. Plotting trajectories using the farther distances shows that the network is unable to adapt to the different distribution of features and has a tendency to generate nonsensical trajectories when tested on them. Note that the compressed version (*CF*) is less sensitive to directly inputting them (*DF*). We expect this is due to the compression learning a basis that is less sensitive to distance-dependent artifacts (even without any explicit penalty in training) and further motivates our use of this architecture.



(a) Compression Type

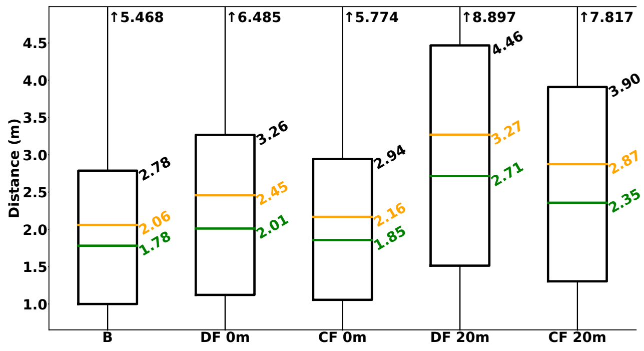


(b) Compression Dimension

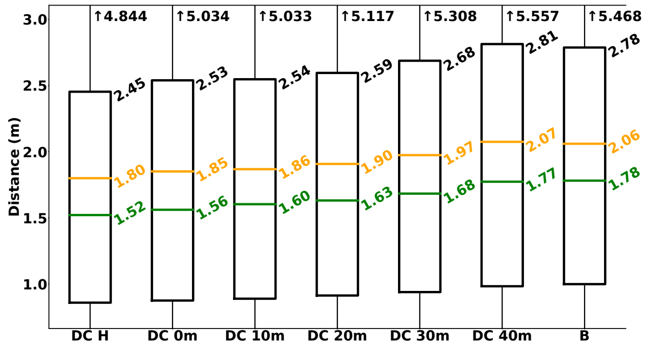


(c) Number Of PCA Features

Fig. 4. Distance error of models at $5s$ using best features in hindsight, *B* is a no-feature baseline, *DF* is directly inputting features into the network, and *C* is compressing features. The model *CF* in 4a, *C 4* in 4b, *C 40* in 4c are all the same and axis are kept consistent between graphs. The green line defines the median, the orange is the mean, the whiskers are defined as ± 1.5 Interquartile Range (IQR) and given by the values with arrows, the other black text is the 75% error. 4a shows the different ways of inputting the features compared to no features. 4b shows the effect of changing the final compression size and the method is robust to this variable. 4c shows the impact of using a variety of PCA features.



(a) Naive Evaluation On Larger Distances



(b) Proposed Approach (DC) On Larger Distances

Fig. 5. Distance error of at $5s$ models on features at varying projection distances. The model *DF* and *CF* are kept consistent from Fig. 4a. *DC* is our proposed distance independent approach. Whiskers are defined as ± 1.5 IQR and given by the black values with arrows, the other black text is the 75% error, the green line defines the median and the orange the mean. Fig. 5a shows that using larger projection distances performs worse than a no-feature baseline *B* with naive training methods. Fig. 5b shows that our approach has improved results with visual features at realistic distances.

Following the approach outlined in Section V-A, our approach (*DC*) is able to handle a variety of distances with lower error than the featureless baseline, as shown in Fig. 5b. When evaluated on hindsight data, training with naively-inputted features and distance-independent training perform similarly, meaning the distance-independent training is learning similar information just on a distance-independent basis. At greater distances, there begins to be significant issues with LiDAR sparsity and occlusions that make the features unreliable, as seen in the high percentage of missing data (see Section V-A), which explains the lack of improvement using visual features beyond 40m. We focus our attention on the features in the range from 0m $\rightarrow$ 30m since this is primarily where the dynamics predictions occur while driving the vehicle. Our method outperforms the featureless baseline at those distances.

VII. Conclusion

We present a hybrid model for vehicle dynamics that incorporates visual features of the terrain to anticipate changes in terradynamics. Our method improves upon the baseline model (dynamics learning without vision) by approximately 10% without significant computational burden. We also provide analyses and ablations of the hyperparameters used in the feature compression network and into the dependence on the distance at which the visual feature is collected to mimic realistic driving conditions. We test our method on an extensive dataset of driving data across various terrains. Future work will explore further aspects such as the distance dependence of the visual features, occlusion handling, and ground plane estimation in vegetative environments.

References

[1] Anelia Angelova, Larry Matthies, Daniel Helmick, and Pietro Perona. Learning and prediction of slip from visual information. *Journal of Field Robotics*, 24(3):205–231, 2007. doi: https://doi.org/10.1002/rob.20179. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20179.

[2] Xiaoyi Cai et al. EVORA: Deep Evidential Traversability Learning for Risk-Aware Off-Road Autonomy. *IEEE Transactions on Robotics*, 40:3756–3777, 2024.

[3] Jiaqi Chen, Jonas Frey, Ruyi Zhou, Takahiro Miki, Georg Martius, and Marco Hutter. Identifying terrain physical parameters from vision - towards physical-parameter-aware locomotion and navigation. *IEEE Robotics and Automation Letters*, 9(11):9279–9286, 2024. doi: 10.1109/LRA.2024.3455788.

[4] Gian Erni, Jonas Frey, Takahiro Miki, Matias Mattamala, and Marco Hutter. Mem: Multi-modal elevation mapping

for robotics and learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11011–11018, 2023.

[5] Jonas Frey, Manthan Patel, Deegan Atha, Julian Nubert, David Fan, Ali Agha, Curtis Padgett, Patrick Spieler, Marco Hutter, and Shehryar Khattak. Roadrunner—learning traversability estimation for autonomous off-road driving. *IEEE Transactions on Field Robotics*, 1:192–212, 2024. doi: 10.1109/TFR.2024.3464369.

[6] Mateus V. Gasparino et al. WayFAST: navigation with predictive traversability in the field. *IEEE Robotics and Automation Letters*, 7(4):10651–10658, October 2022. ISSN 2377-3774. doi: 10.1109/lra.2022.3193464. URL http://dx.doi.org/10.1109/LRA.2022.3193464.

[7] Gabrielle Hedrick, Nicholas Ohi, and Yu Gu. Terrain-aware path planning and map update for Mars sample return mission. *IEEE Robotics and Automation Letters*, 5(4):5181–5188, 2020. doi: 10.1109/LRA.2020.3005123.

[8] Sanghun Jung, JoonHo Lee, Xiangyun Meng, Byron Boots, and Alexander Lambert. V-STRONG: visual self-supervised traversability learning for off-road navigation, 2024.

[9] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019. doi: 10.1109/LRA.2019.2926677.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

[11] Wenjun Liu, Yulin Zhai, Guang Chen, and Alois Knoll. Gaussian process based model predictive control for overtaking scenarios at highway curves. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1161–1167, 2022. doi: 10.1109/IV51971.2022.9827233.

[12] Elena Sorina Lupu, Fengze Xie, James Alan Preiss, Jedidiah Alindogan, Matthew Anderson, and Soon-Jo Chung. Magicvfm-meta-learning adaptation for ground interaction control with visual foundation models. *IEEE Transactions on Robotics*, 41:180–199, 2025. doi: 10.1109/TRO.2024.3475212.

[13] Matías Mattamala et al. Wild visual navigation: Fast traversability learning via pre-trained models and online self-supervision. *arxiv.org:2404.07110*, 2024. URL https://arxiv.org/abs/2404.07110.

[14] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

[15] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11138–11147, 2020.

[16] Amirreza Shaban, Xiangyun Meng, JoonHo Lee, Byron Boots, and Dieter Fox. Semantic terrain classification for off-road autonomous driving. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 619–629. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/shaban22a.html.

[17] Bogdan Vlahov, Jason Gibson, Manan Gandhi, and Evangelos A. Theodorou. MPPI-Generic: a CUDA library for stochastic optimization, 2024. URL https://arxiv.org/abs/2409.07563.

[18] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018. doi: 10.1109/TRO.2018.2865891.

[19] Hanzhang Xue et al. Contrastive label disambiguation for self-supervised terrain traversability learning in off-road environments. *arxiv.org:2307.02871*, 2023.