Appendix for Velociraptor: Leveraging Visual Foundation Models for Label-Free, Risk-Aware Off-Road Navigation

Samuel Triest[1], **Matthew Sivaprakasam**[1], **Shubhra Aich**[1], **David D. Fan**[2], **Wenshan Wang**[1], **Sebastian Scherer**[1]
[1]Carnegie Mellon University Robotics Institute, [2]Field AI
striest@andrew.cmu.edu

A Implementation Details

A.1 Platform

We test our method using a full-scale Yamaha Viking VI modified for autonomous driving by Mai et al. [1] and Sivaprakasam et al. [2]. This platform contains a front-facing Multisense stereo camera and Velodyne LiDAR, and is capable of autonomous speeds of up to roughly $10m/s$. In order to register scans and maps together, we use Super Odometry [3]. Our local maps are centered on the robot in the odometry frame and represent an area of $120m \times 120m$ ($60m$ forward) at $0.5m$ per cell.

A.2 Geometric Feature Extraction

A list of geometric features extracted is presented in Table 1. Our terrain estimation algorithm works by computing the minimum elevation for each cell containing points. This elevation is then interpolated and smoothed using a Markov Random Field [4].

A.3 Visual Feature Extraction

For visual feature extraction, we leverage several state-of-the-art neural network backbones. For semantics, we leverage the open-source implementation of GA-Nav [5], a state-of-the-art FPV model for off-road semantic segmentation. Images for GA-Nav are resized to $[688 \times 550]$. For Dinov2 [6], we leverage values (as opposed to query/key) from the tenth layer of the ViT-b backbone. This was chosen via qualitative analysis of the features and the guidance from AnyLoc [7], as well as runtime constraints. For Segment Anything, we use the image features provided by the open-source API [8]. For both VFM-based methods, images are resized to $[686 \times 364]$. Similar to prior work Emernerf [9], we perform a Principal Component Analysis (PCA) on the visual feature embeddings in FPV-space. The number of features was reduced to sixteen to fit memory constraints for on-board local mapping, and roughly match the number of semantic channels from GA-Nav (). Image features to be included in the PCA analysis are randomly selected from pixels in the train set which have a corresponding LiDAR return. From this collection of features, we can then perform a PCA decomposition (Equation 1). At test time, the PCA reduction can be computed efficiently per-pixel via Equation 2.

Instantaneous visual maps M_t are computed by first projecting the point cloud into the image frame using camera intrinsics K and extrinsics $[R|t]$ (Equation 3). Thus results in a feature-colorized pointcloud $\mathbf{P}$, where each point p has a position $p_{x,y,z}$ and feature p_f. All visual features with a corresponding point are then projected into BEV with the spatial locations of their corresponding points, given local map parameters (origin, length and resolution). Cells with multiple points have their embeddings averaged (Equation 4). Maps are aggregated over time by first transforming the

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

previous map $\mathbf{M}_{t-1}$ to the current pose x_t and then combining with M_t using an exponential moving average (Equation 5) to generate an aggregated visual map $\mathbf{M}_t$ centered at x_t.

$$\mu_{PCA} = \frac{1}{n}\sum_i [f_i], \quad U, S, V^T = SVD(F - \mu_{PCA}) \tag{1}$$

$$f_{PCA} = (f - \mu_{PCA})V^T \tag{2}$$

$$[u, v, w]^T = \mathbf{K}[\mathbf{R}|\mathbf{t}][x, y, z, 1]^T, \quad u = \frac{u}{w}, \quad v = \frac{v}{w} \tag{3}$$

$$M_t^{(i,j)} = \frac{1}{|\mathbf{P}_t^{i,j}|}\sum_{p \in \mathbf{P}_t^{i,j}} p_f, \quad \mathbf{P}_t^{i,j} = \{p \in \mathbf{P}_t | p_{x,y,z} \in M_t^{i,j}\} \tag{4}$$

$$\mathbf{M}_t^{i,j} = \begin{cases} M_{t-1}^{i,j} & \nexists M_t^{i,j} \\ M_t^{i,j} & \nexists M_{t-1}^{i,j} \\ \alpha M_t^{i,j} + (1-\alpha) M_t^{i,j} & \text{otherwise} \end{cases} \tag{5}$$

Table 1: Qualitative Description of the Geometric Terrain Features

Feature	Description
terrain	The output of our terrain estimation algorithm.
terrain slope	The magnitude of the slope of the terrain estimate
diff	The difference between the maximum elevation of a cell and its terrain estimate
SVD_{1-3}	The principal components of the points in each grid cell
roughness	The variance in elevation of the points in each cell
unknown	Binary indicator of whether there are measurements for each cell
VF_{1-N}	The features from the visual mapping (DINO, SAM, GA-Nav)

Our mapping pipeline is able to run at $10hz$ on a laptop with a 13th gen Intel i9 CPU and Nvidia 3080 Laptop GPU.

A.4 Model-Predictive Controller

We use MPPI [10] to operate on our learned representation. We use a modified version of MPPI that leverages Gaussian Random Walks and an action library, and a kinematic bicycle model with additional steering and throttle dynamics (Equation 6, $a(x, u), d(x, u)$ were nonlinear functions that we fit on a small sysid dataset, following the equation forms from Mai et al. [1]). Our implementation of MPPI attempts to minimize a weighted sum of distance to goal, and vehicle footprint projection onto the costmap (Equation 7). The speedmap is treated as a constraint with some shaping, and speeds that exceed the value in the speedmap are assigned infinite cost (Equation 8). Values in the uncertainty map are treated as obstacles and also assigned infinite cost (note that this means there is no need to weight the uncertainty map in the cost function). In practice, we also include a footprint (treated as a fixed set of poses calculable from robot state) for the map terms to account for the fact that the vehicle will reside in multiple cells. Over the footprint, we take an average for costs and a minimum for speeds, which is applied for both training and deployment.

$$X = \begin{bmatrix} x \\ y \\ \psi \\ v \\ \delta \end{bmatrix}, \quad U = \begin{bmatrix} t \\ \delta_{des} \end{bmatrix}, \quad \dot{X} = \begin{bmatrix} vcos(\psi) \\ vsin(\psi) \\ vtan(\delta)/L \\ a(x, u) \\ d(x, u) \end{bmatrix} \tag{6}$$

Table 2: MPPI parameters

Parameter	Value	Description
H	50	number of timesteps
dt	0.1s	timestep discretization
K_1	0.1	goal weight
K_2	1.0	cost weight
K_3	1.0	speed weight
num actlib samples	100	number of action library samples
num random walk samples	512	number of random walk samples
Footprint	$4m \times 2m$	vehicle footprint

$$J(\tau, C, S) = K_1||p(x_T) - g||_2 + \sum_t \left[\frac{1}{|F(x_t)|} \sum_{p_t \in F(x_t)} [K_2 C(p_t) + K_3 S'(p_t, v(x_t)) + U(p_t)] \right] \quad (7)$$

$$S'(p_t, v_t) = \begin{cases} exp(||v_t||_2 - S(p_t)), & \text{if } ||v_t||_2 < S(p_t) \\ \infty, & \text{otherwise} \end{cases} \quad (8)$$

A.5 Computation of Training Signals

In order to train our networks, we must compute discrete map-space signals from continuous-valued trajectories. Namely, we must produce state visitation distributions D and speed label maps $\hat{S}$, where $D(\tau)$ contains the fraction of τ spent in each cell, and $\hat{S}(\tau)$ contains the average speed that τ traversed each cell at. D is used directly in the IRL objective, and $\hat{S}$ can be masked to compute the speedmap objective. At a high level, this is done by mapping each state to grid coordinates and averaging (for state visitations, we can scatter any constant value, and for the speedmap, we can scatter the expert speed). A more thorough explanation is provided in Algorithm 1. Following guidance from LEARCH [11], we apply footprint expansion (represented as a function $F(x)$ that returns a set of positions in the footprint, given state x) to the computation of state visitation maps and speed label maps.

B Dataset Details

Our dataset was collected over the span of several months at two testing sites (site 1 and site 2) by two expert drivers. Data collection was focused specifically focusing on particular areas in the site containing challenging terrain (e.g. steep slopes, dense vegetation, etc). Drivers were instructed to drive as quickly as possible while remaining safe. Runs from site 1 were then partitioned into the Train and Test 1 datasets (partitioning was done at the run level to make Test 1 require more generalization). We chose to put the most challenging scenarios in Test 1 to stress the ability of Velociraptior to handle challenging terrain. The Test 2 dataset was created using all the runs from site 2 (e.g. Test 2 represents somewhat out-of-distribution data). All runs in the datasets were then manually downselected to focus on the most interesting terrain interactions.

Each run consists of a timestamped sequence of maps $\{\mathbf{M}\}$ and states $\{x\}$. Note that the timestamps and number of elements of these data may not match. We generate samples for our dataset $\mathcal{D}$ via the procedure in Algorithm 2 given a run $(\{\mathbf{M}\}, \{x\})$, time discretization dt, planning horizon H, and spacing k (in our case $k = 2s$). For ease of notation, let $t(x)$ return the time at which data x was collected.

That is, every k frames, we add to our dataset the current map representation $\mathbf{M_t}$ and next H steps of expert trajectory. The learner will start at the current state x_t and must reach the state H steps in the future x_{t+H}. This process is repeated for each unique run in the dataset.

Algorithm 1: Computation of $D(\tau)$ and $\hat{S}(\tau)$

Input:
Trajectory τ,
Footprint expansion function $F(x)$,
Grid discretization function $G(x)$,
Map size (W, H),

```
D = 0^{W×H}                          ◁ initialize state visitation map
Ŝ = 0^{W×H}                          ◁ initialize speed label map
for x ∈ τ do
    X = F(x)                         ◁ apply footprint expansion to state
    s = ||v(x)||_2                   ◁ get speed at current state
    for x' ∈ X do
        i, j = G(x')                 ◁ get grid indices
        if i ∈ [0, W) and j ∈ [0, H) then
            D_{i,j} = D_{i,j} + 1    ◁ add to state visitations
            Ŝ_{i,j} = Ŝ_{i,j} + s    ◁ add to speed labels
        end
    end
end
Ŝ = Ŝ / D                            ◁ normalize speeds
D = D / Σ D                          ◁ normalize probabilities
return D, Ŝ
```

Algorithm 2: Dataset Generation

Input:
Timestamped map data $\{\mathbf{M}\}$,
Timestamped state data $\{x\}$,
Time discretization dt,
Horizon H,
Spacing k

```
t_0 = −∞                             ◁ initialize start time
𝒟 = ∅                                ◁ initialize dataset
Create trajectory interpolator X : t → x ∈ SE(3) from {x}
for M ∈ {M} do
end
if t(M) − t_0 > k then
    T = [t(M), t(M) + dt, ..., t(M) + H * dt]   ◁ get target times
    τ^E = X(T)                       ◁ get trajectory from interpolator
    x_0 = τ^E_0                      ◁ set initial state
    x_g = τ^E_H                      ◁ set goal state
    𝒟 = 𝒟 ∪ (M, τ^E, x_0, x_g)       ◁ add to dataset
    t_0 = t(M)                       ◁ set last updated time
end
return 𝒟
```

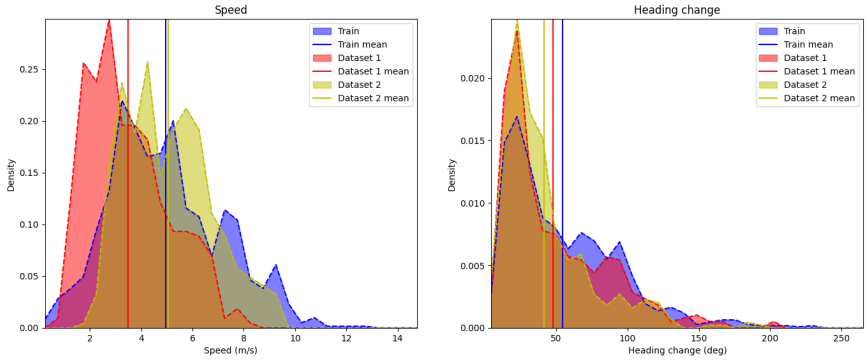

Figure 1: A quantitative analysis of our datasets.

Table 3: Dataset Description

Dataset	Num samples	Num unique runs	Avg. Speed	Avg. Turn	H	K
Train	1209	9	$4.96m/s$	54.48 deg	100 ($10s$)	$2s$
Test 1	429	5	$3.49m/s$	47.71 deg	100 ($10s$)	$2s$
Test 2	634	8	$5.05m/s$	41.42 deg	100 ($10s$)	$2s$

Our dataset statistics are presented in Table 3. Additional qualitative visualizations of the terrain in the datasets is presented in Figure 2. We report some additional trajectory-level analysis of the three datasets in Figure 1. We choose to report the average speed of each trajectory and the total change in heading as these metrics roughly describe the speed and complexity of the trajectories that the learner must reproduce. Overall, we find that all three datasets are relatively similar in terms of heading change, while Test 1 has a lower average speed. This is expected, as the scenarios in Test 1 were hand-selected to be the most challenging, and thus required the expert to drive more slowly compared to the other datasets.

C Training Details

Parameters for our training procedure, network architecture and uncertainty estimation are presented in Tables 4, 5 and 6, respectively. Note that in order to reduce memory consumption, our ensemble is represented by creating E copies of the first FCN layer, and sharing the remaining parameters.

Figure 2: A qualitative comparison of the test environments. (Top row) Dataset 1 contains open scenarios, slopes, trails and seasonal variation. (Bottom row) Compared to Dataset 1, Dataset 2 has more grassland terrain, vegetation and open slopes.

Table 4: Training parameters

Parameter	Value	Description
Epochs	5	number of epochs
Batch Size	4	batch size
Speed coeff	1.0	weight of speed learning and cost learning objectives
Cost Regularization	10^{-8}	L2 cost regularization coefficient
MPPI iterations	10	number of mppi iterations to converge
Optimizer	Adam [12]	optimizer

Table 5: Network Parameters

Parameter	Value	Description
E	16	ensemble size
s_{max}	$15m/s$	maximum speed
s_n	15	number of speed bins
hidden sizes	[128, 128]	intermediate feature map sizes
hidden activation	tanh	activation type
num params per ensemble	315364	number of learnable parameters for a single ensemble member
num params total	903474	total number of learnable parameters across the ensemble

D Additional Quantitative Results

D.1 Overfitting Experiment

In order to verify that our network is able to generalize its performance to novel terrains, we compare the performance of **V-Dino** across the Train, Test 1 and Test 2 datasets. We report the positional and speed MHD metrics (pMHD, sMHD), per the main paper. In order to more directly compare metrics across datasets, we additionally report positional MHD divided by the average speed of each dataset (pMHD/speed). This is done because MHD is proportional to trajectory scale, which is itself (roughly) proportional to speed. Results are presented in Table 7. Overall, we can observe that V-Dino does not seem to significantly overfit to the training dataset, as evidenced by relatively similar metrics across all datasets. We suspect that despite the relatively small size of our dataset, we are able to avoid significant overfitting because: 1) our dataset collection method promotes diversity of samples, 2) we use a relatively lightweight FCN, and 3) our network is fed meaningful features that promote better generalization than raw sensor data. However, we do not claim that this trend will continue for more out-of-distribution terrains (such as deserts, mountains), which may require additional (label-free) data collection.

D.2 Feature Entropy Experiment

We additionally evaluate the stability of our learned features across the learned methods. We report in Table 8 the variance of the features from the FCN backbone across the ensemble members, over the two evaluation datasets. We observe that in general, the foundation model features result in lower feature variance, meaning that the cost and speed predictions are more stable, and better explain the expert trajectories.

Table 6: Uncertainty Parameters

Parameter	Value	Description
num clusters	8	number of visual feature clusters
distance function	cosine	distance metric for clustering
uncertainty threshold	.9	min value of residual to be considered uncertain

Table 7: Comparison of V-Dino metrics across train and test sets

V-Dino	pMHD	pMHD/speed	sMHD
Train	1.31 ± 0.03	0.26 ± 0.00	0.18 ± 0.00
Test 1	0.83 ± 0.03	0.24 ± 0.00	0.14 ± 0.00
Test 2	1.48 ± 0.02	0.29 ± 0.00	0.15 ± 0.01

Table 8: Feature variance of Velociraptor variants over the evaluation datasets

	Test 1	Test 2
V-geom	0.2649 ± 0.0177	0.2544 ± 0.0186
V-semantics	0.2646 ± 0.0127	0.2978 ± 0.0153
V-SAM	$\mathbf{0.2232 \pm 0.0134}$	0.2508 ± 0.0129
V-Dino	0.2351 ± 0.0053	$\mathbf{0.2380 \pm 0.0094}$

E Additional Qualitative Results

E.1 Overview and Analysis of Navigation Trials

We present in Figure 3 a GPS overview of the navigation course. Additional details relating to the speed of each method are presented in Figures 4 and 5. Overall, we find that Velociraptor is able to successfully spend more time at higher speeds while still reducing overall speed when needed. In particular, we observe that geometric features alone are not sufficient to perform appropriate speed selection. This is indicated by the geometric baseline's low speed on open trails (Figure 4), and overall bunching of speeds around 6m/s (Figure 5). While the semantics-based method is able to more intelligently modulate robot speed, we find that it is overly conservative with respect to vegetation, resulting in lower overall speed. Velociraptor is able to safely spend more time at higher speeds (7+ m/s) with fewer interventions. The uncertainty estimation layer, while adding safety, reduces the overall speed somewhat as there are additional obstacles that the robot must avoid (thus resulting in lower speed). However, both variants of Velociraptor are able to spend more time at 7+ m/s than the geometric and semantic baselines.

E.2 Corrections from the Navigation Trials

Provided in Figure 6 is a representative set of scenarios that led to corrections from our method. Overall, we believe that many sources of corrections can be avoided by improving our on-board

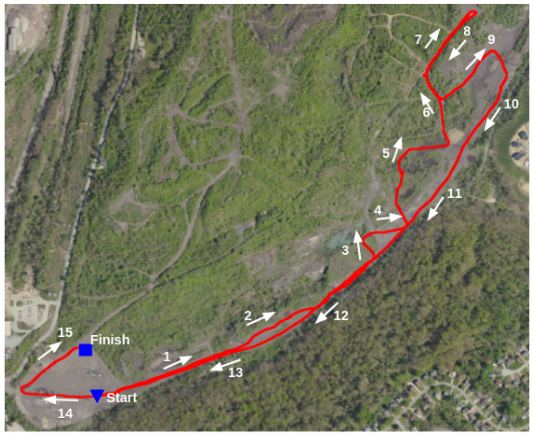

Figure 3: An Overview of the large-scale navigation course. The robot started at the blue triangle, and had to follow a series of waypoints, roughly following the ordering of the arrows (the arrows do not denote the specific waypoints). This loop contains many challenging traversability scenarios such as steep slopes, tall grass, standing water, and small, out-of-distribution obstacles.

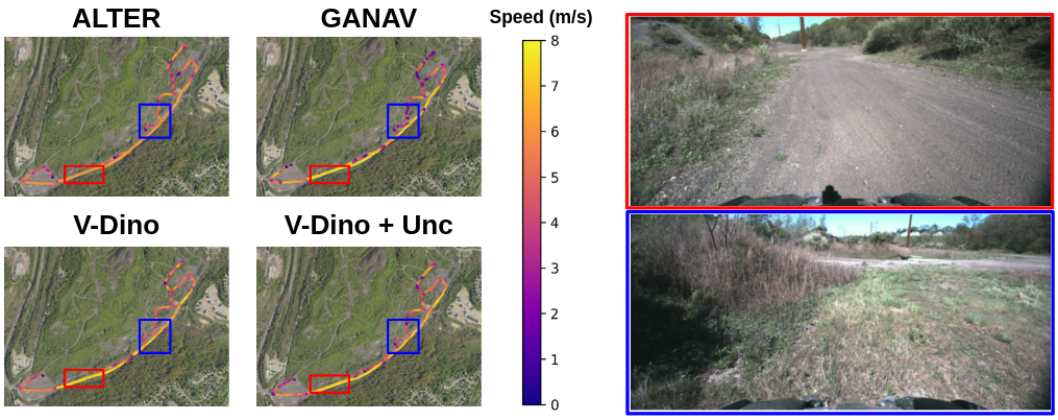

Figure 4: Overview of the navigation trials, colored by speed. Highlighted are two scenarios where V-Dino outperforms existing baselines. (Red) Geometric features alone are not sufficient to distinguish trails from other terrains, causing the geometric baseline to drive more slowly than other methods on open trails. (Blue) Semantic classes are not expressive enough to distinguish between different types of grass, causing the semantic baseline to be overly conservative and slow when the vehicle must drive in/through vegetation.

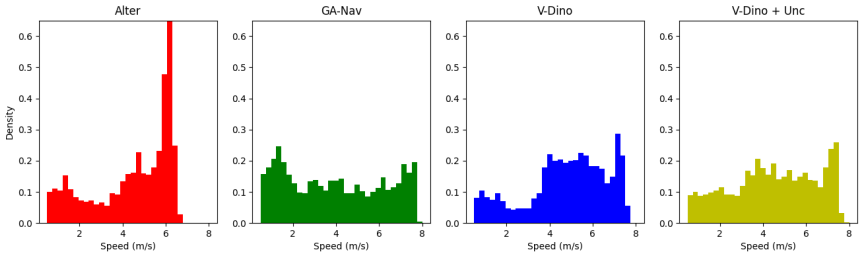

Figure 5: Speed distributions across the four methods on the large-scale navigation trial.

planning and control. Specifically, we can handle small ditches and other negative obstacles with improved terrain estimation techniques and higher resolution maps, we can handle cul-de-sacs with improved planning methods, and we can handle uncertainty false positives by leveraging the geometric and visual features together.

E.3 Examples from the Test Dataset

Provided in this section are additional qualitative results that highlight the differences between each method (Figs. 7 and 8). Figure 7 demonstrates a scenario in which the trail is obscured by tall vegetation. This vegetation results in both the geometric and semantic analysis underperforming. However, our continuous-valued features are able to disambiguate the trail, resulting in better costing. In Figure 8, we are in a scenario in which there is multiple grass types. The grayer grass is marshy and should be avoided if possible. The features from Dino are best able to capture the intra-class variation, resulting in better costing for IRL-Dino, as compared to geometry-only and semantic features.

F Additional Future Work

While we present results in a previously unseen environment, due to logistical limitations, the environment is fairly similar to the environment in which the training data was collected (e.g. highly vegetated, hilly terrain). Further experimentation is required to assess Velociraptor's capability to generalize to more varied terrain (e.g. deserts, mountainous terrain, etc.). Such experiments can

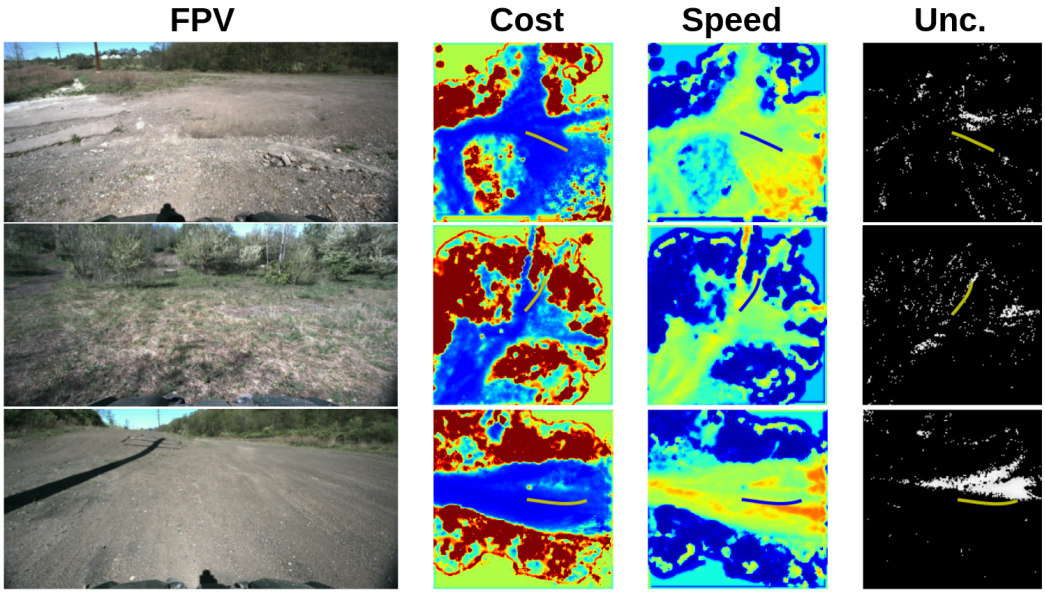

Figure 6: Some qualitative analysis of corrections from the navigation trials. (Top) A small ditch was not perceived by our geometric feature extractor, leading Velociraptor to be overly optimistic about the terrain in front. (Middle) The planning horizon of MPPI was not sufficient to observe a large cul-de-sac. (Bottom) Shadows adversely affected the uncertainty estimation (as it relies purely on vision). Note that this particular instance did not result in a correction.

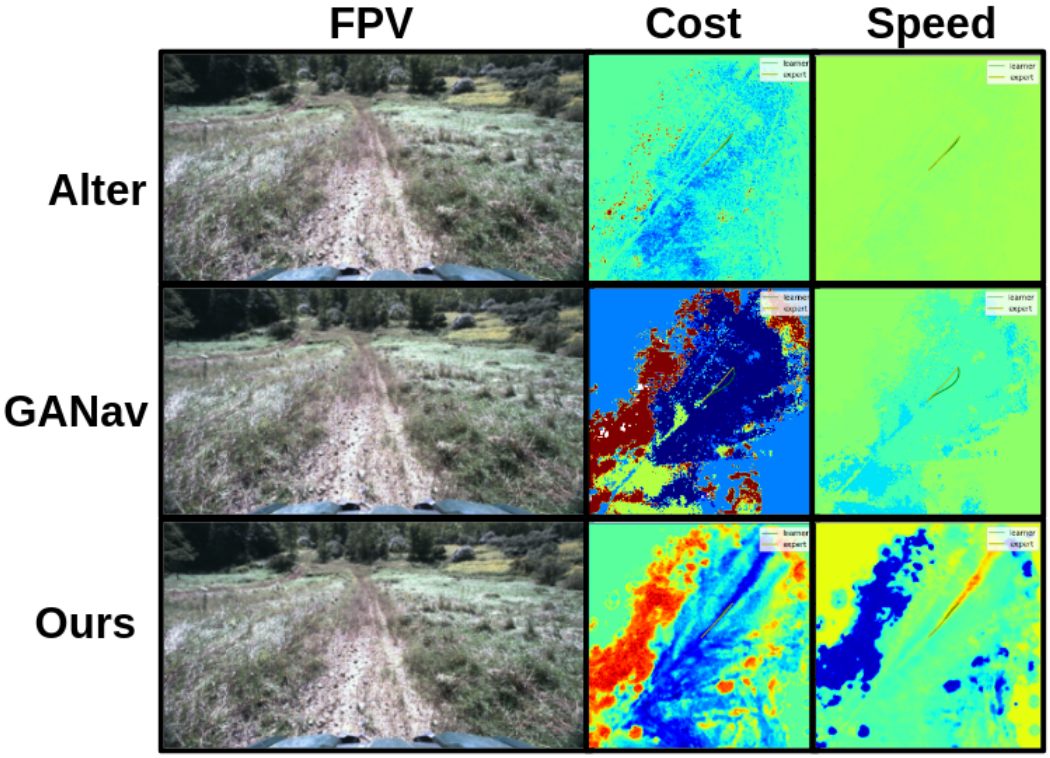

Figure 7: An example from our test set. The trail in front is obscured by grass, negatively impacting the performance of the geometric and semantic baselines. However, our method is able to differentiate between the trail and surrounding vegetation.

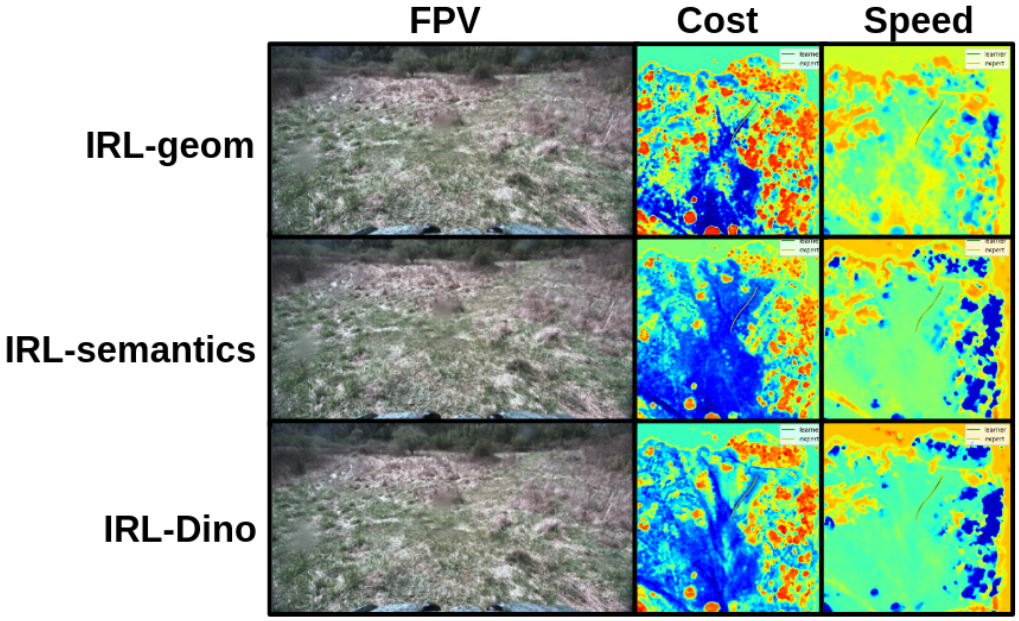

Figure 8: Another example from our test dataset. The brown grass contains marshier soil and was avoided when possible in the train set. IRL trained with only geometric features is not able to fully assign low cost to the grass. IRL trained with semantic features is not able to differentiate between the grass types. IRL trained with Dino features can differentiate between the grass types.

include evaluating the importance of adapting the Dino PCA, assessing the generalization of the network trained in one biome to another, and the amount of expert data required to achieve sufficient performance in a novel environment. Additionally, we would like to demonstrate this system's capability on additional platforms.

References

[1] J. Mai. System design, modelling, and control for an off-road autonomous ground vehicle. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, July 2020.

[2] M. Sivaprakasam, P. Maheshwari, M. G. Castro, S. Triest, M. Nye, S. Willits, A. Saba, W. Wang, and S. Scherer. Tartandrive 2.0: More modalities and better infrastructure to further self-supervised learning research in off-road driving tasks. *arXiv preprint arXiv:2402.01913*, 2024.

[3] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer. Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8729–8736. IEEE, 2021.

[4] D. Maturana. *Semantic Mapping for Autonomous Navigation and Exploration*. PhD thesis, Carnegie Mellon University, 2022.

[5] T. Guan, D. Kothandaraman, R. Chandra, A. J. Sathyamoorthy, K. Weerakoon, and D. Manocha. Ga-nav: Efficient terrain segmentation for robot navigation in unstructured outdoor environments. *IEEE Robotics and Automation Letters*, 7(3):8138–8145, 2022.

[6] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[7] N. Keetha, A. Mishra, J. Karhade, K. M. Jatavallabhula, S. Scherer, M. Krishna, and S. Garg. Anyloc: Towards universal visual place recognition. *IEEE Robotics and Automation Letters*, 2023.

[8] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[9] J. Yang, B. Ivanovic, O. Litany, X. Weng, S. W. Kim, B. Li, T. Che, D. Xu, S. Fidler, M. Pavone, et al. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. *arXiv preprint arXiv:2311.02077*, 2023.

[10] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[11] N. D. Ratliff, D. Silver, and J. A. Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.