Online-Adaptive Self-Supervised Learning with Visual Foundation Models for Autonomous Off-Road Driving

Matthew Sivaprakasam

CMU-RI-TR-24-57

August 5, 2024

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Prof. Sebastian Scherer, *chair*
Dr. Wenshan Wang
Samuel Triest

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To Tsaocaa. What is a master's degree without all the little treats along the way?

Abstract

Autonomous robot navigation in off-road environments currently presents a number of challenges. The lack of structure makes it difficult to handcraft geometry-based heuristics that are robust to the diverse set of scenarios the robot might encounter. Many of the learned methods that work well in urban scenarios require massive amounts of hand-labeled data, but the nuances of deciding where a robot can and cannot drive in off-road terrain make it difficult to label large-scale data the same way. Many state-of-the-art approaches instead leverage self-supervised methods in training, using either expert demonstrations or proprioceptive feedback, but often still require a lot of data and can be vulnerable to domain shifts.

We adopt a philosophy that learned methods for off-road driving should be both self-supervised and adaptive, such that the robot can learn online without a human in the loop. In this work we propose a method that leverages proprioceptive cues and pre-trained visual foundation models to rapidly adjust its understanding of its environment in real-time, eliminating the need for large-scale training data and hand-labels. Specifically, we introduce a framework that predicts costmaps, speedmaps, and uncertainty by associating incoming visual features with roughness experienced by the system. Within seconds of collected experience, our results demonstrate navigation performance with as few interventions as methods trained on 100-1000x more data, while travelling as quickly as possible within the constraints of rider comfort. Furthermore, we aim to reduce the barrier to entry to full-scale off-road driving research by presenting TartanDrive 2.0, a large multi-modal dataset geared towards self-supervised learning methods.

Acknowledgments

First of all, I would like to thank my committee: Dr. Sebastian Scherer, Dr. Wenshan Wang, and Samuel Triest. Basti and Wenshan took me as an undergraduate student despite my lack of robotics experience. It was thanks to their guidance and mentorship, as well as the supportive environment that they work to maintain in the lab, that I was able to so quickly start contributing to state-of-the-art research and learn about aspects of robotics that I never new existed. I would also like to thank Sam, not only for his invaluable mentorship but also as someone who has been an integral part of the work developed in this thesis. From software all the way through hardware integration and field testing this has all been a team effort. Speaking of team, there are so many others who I've been privileged to work with on the off-road driving projects, specifically Shubhra Aich, Parv Maheshwari, Mateo Guaman Castro, Anton Yanovich, Micah Nye, and Steve Willits. Additionally, I'd like to thank Humeyra Kacar, Jeric Lew, Isaiah Adu, and Joshua Pen, not only for their valuable contributions but also for putting up with me as a mentor as I learned how to lead. Finally, thank you to our collaborators at ARL, specifically Jason Gregory, John G. Rogers, Felix Sanchez, Ian Lancaster, and Christa.

There are also a number of other people in the robotics community who have played a vital role in my journey as a researcher. Andrew Saba and Joshua Spisak in many ways paved a clear path for me to get involved at CMU, helping me every step of the way. More than that, they showed me the importance of paying it forward and working to uplift others the same way they did for me. I'd also like to thank my undergraduate advisor Dr. Samuel Dickerson for the opportunities I was provided at Pitt due to his involvement in the computer engineering program, and Rachel Burcin and Dr. John Dolan for fostering the RISS community that I've been fortunate to be a part of. Additionally I'd like to shout out Brady Moon and Cherie Ho for teaching me how to make better presentations through storytelling and visuals. Finally, the past two years would have been much more grueling without the friends in the RI I've made along the way: Sam Schoedel, Sofia Kwok, Conner Pulling, Nayana Suvarna, John Zhang, Jay Karhade, Nikhil Keetha, and many others. Thanks for helping me keep the perfect balance of being distracted and productive in the lab, and for the fun times elsewhere.

Having lived in Pittsburgh over the past seven years now, there are several people outside of CMU who I would like to mention for helping me get to where I am today. Thank you to Andrew Ashley for his never-ending support and for not getting tired of me despite never having lived more than two doors away from me since we both moved to Pittsburgh. Thank you to all of the friends I've made while at Pitt who still keep me sane today: Bernie, Chaim, Kent, Erin, Reni, Shiv, Pablo, Dan, Prem, Tom, Ro, Robin, Miriam, and more - too many to count.

Most importantly, I would like to thank my family. I would not have made it through this program, or even gotten into it to begin with, without their support. They have been with me the whole way through, from back in high school and early college when I couldn't decide what I wanted to do with my life to finally settling on a career path that I am truly passionate about. Thanks to them I've accomplished things I'd never imagined in my future even as recent as a couple years ago. As much as you like to say that my work is "all me", it is a reflection of the experiences, opportunities, and advice that you have given me.

Funding

This work was supported by ARL awards #W911NF1820218 and #W911NF20S0005.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Off-road autonomous driving is becoming an increasingly researched topic due to its wide range of applications. Robots are already being deployed in fields such as agriculture, search and rescue, construction, and defense, where they are expected to operate in unstructured and diverse environments. In order to perform robustly, they must be able to reason about terrain with little to no structure or markings and navigate from goal to goal without crashing or getting stuck.

This task is not necessarily specific to off-road driving with extreme ruggedized vehicles and robots. From beaten dirt roads in rural areas to chaotic intersections, potholes, and constant construction in cities such as Pittsburgh, human drivers are constantly presented with scenarios in which they must deviate from the default rules and structure of day-to-day driving and make their own informed decisions. For a robot to navigate anywhere in the real world, it must be able to do the same.

Research in off-road autonomy even has impact beyond driving itself, as many of the underlying challenges exist in numerous other areas. The development of a system that can quickly learn from its own experiences to obtain a holistic understanding of its environment, which this thesis aims to take a step toward, is a common goal across many fields of robotics. The off-road domain simply provides a medium through which we can tackle some of the problems that make robotics as a whole difficult.

1.2 Background and Task Definition

1.2.1 What is Autonomous Off-Road Driving?

While off-road driving is often referred to in the context of trail driving specifically, for the purpose of this thesis we expand the term to include any off-road terrain that a given robot could reasonable be expected to traverse, regardless of the presence of a trail (Fig. 1.1). An autonomous off-road system should be able to handle trails in various conditions such as loose dirt, thick mud, and snow, but it should also be robust to complex scenarios in which there aren't any marked paths such as navigating through tall grass while avoiding boulders and trees. Moreover, it should be able to navigate in a way that allows it to reach its goal quickly and with minimal damage and wear.

1.2.2 Problem Formulation

As is common in much of the state-of-the-art [1, 2, 3, 4, 5], we define the off-road driving task as a trajectory optimization problem. The robot must navigate a large distance from point A to point B via a set of waypoints $w_{1:N}$ while optimizing the objective:

$$\begin{aligned} \min_{u_{1:T-1}} \quad & J(x_{1:T}, u_{1:T-1}, w_n) \\ \text{s.t.} \quad & x_{t+1} = f(x_t, u_t) \end{aligned} \tag{1.1}$$

in which the states $x_{1:T}$ are determined by the actions $u_{1:T-1}$ chosen and rolled out through dynamics model f. In our formulation, the main behaviors we care about are avoiding treacherous terrain, going as quickly as possible, and reaching each waypoint. In order to implement these in a way that is interpretable, we generate a map representing the space around the vehicle, with a cost and desired speed assigned to each cell. This results in the cost function J with three terms:

$$J(x_{1:T}, u_{1:T-1}, w_n) = k_c \sum_{t=1}^{T} J_c(x_t) + k_s \sum_{t=1}^{T} J_s(x_t) + k_w \left\| p(x_T) - w_n \right\|_2 \tag{1.2}$$

where J_c uses the state to query the 2D *cost* values from the map dictating the traversability of terrain around the vehicle in metric space, J_s represents the 2D *speed* values from the map, and function p converts the state into the same frame as *waypoint* w_n. k_c, k_s, k_w are scalars that allow us to weight each cost differently. The waypoints provide high-level direction while the costmap and speedmap provide local information necessary to avoid treacherous terrain. By including both cost and speed in the metric space we enable the robot to know not only *where* to drive but also *how*, for example driving at high speeds on trails and slowing down in tall grass.

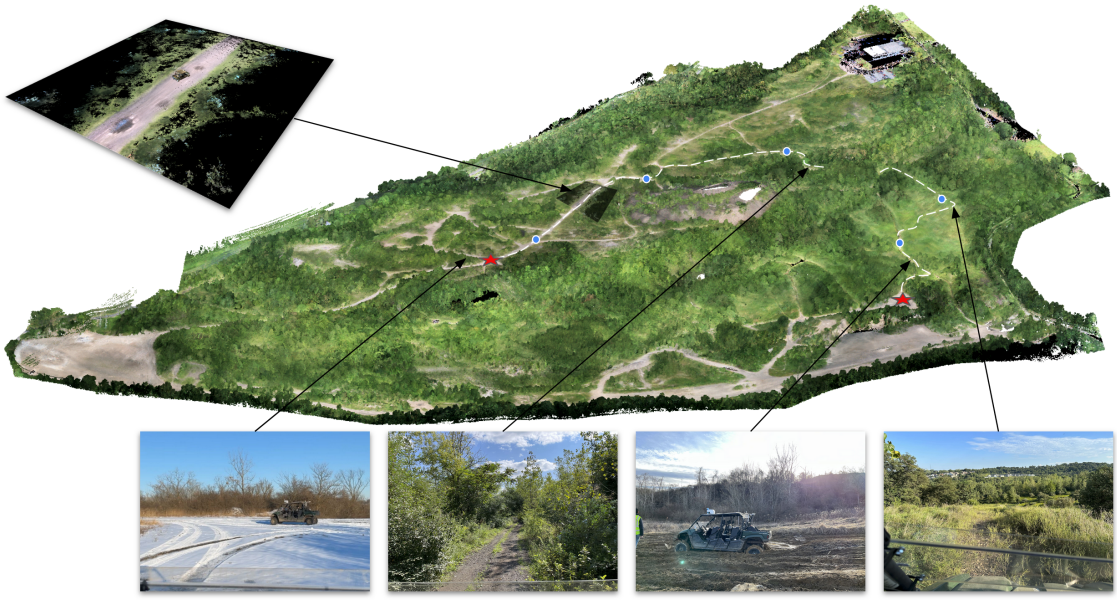

Figure 1.1: Overview of the off-road driving problem, which includes navigating not just trails but complex scenarios such as mud, hills, and unmarked paths (bottom right). The system is given a set of sparse waypoints (blue circles) to navigate from point A to point B. Local information is represented in a top-down space (top left) such that a local planner can detect and avoid treacherous terrain.

1.3 Challenges

Much recent research has focused primarily on improving navigation at the local planning level, in which the challenges of complex unstructured terrain are dealt with at a finer level of detail. Obstacles such as rocks and trees must be detected without being conflated with traversable terrain. This task is made even more difficult

by sensor limitations and the need for real-time decision-making. Ensuring robust performance across a diverse set of environments that themselves inherently change over time requires advanced and adaptive perception.

Costmap generation based on methods such as height-thresholding [6] and semantic segmentation [7, 8] often struggle in this domain because they rely on basic assumptions about the terrain, such as that all terrain above a certain height is non-traversable and that all terrain within the same semantic class has the same physical properties. There have been extensions, such as those that perform more sophisticated geometric analysis to produce a cost [5, 9, 10], but can require extensive hand-tuning, be sensitive to sensor and odometry noise, and fail to reason about different terrain with similar geometry. Learned methods have shown potential to address these issues [1, 2, 4, 11] but are trained offline and require large amounts of hand-labeled data, introducing scalability problems.

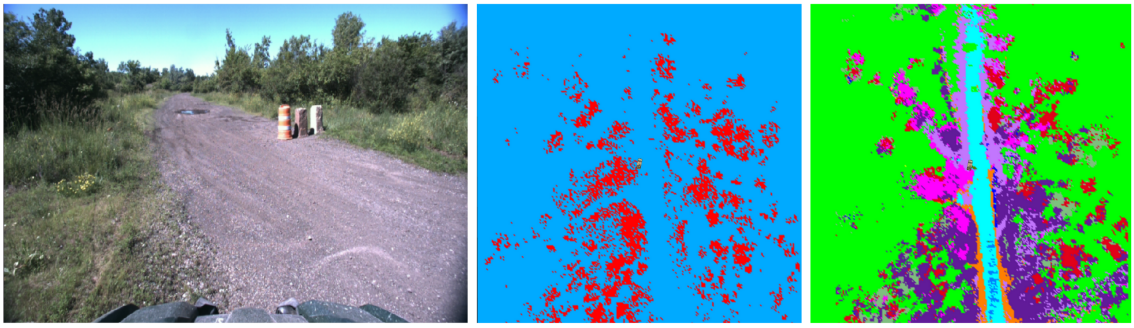

Figure 1.2: Many state-of-the-art approaches to autonomous off-road driving employ height-based (middle) or semantic costs (right), neither of which capture the variations in terrain at a high level of detail.

Many of these challenges and shortcomings can be addressed by furthering the state-of-the-art in two areas:

- **Self-supervision** enables the system to learn from its own experiences without requiring expensive human-labeled data, allowing for easier scalability. This requires strong proprioceptive signals and information that relate a robot's actions to the cost of its resulting experiences.
- **Adaptation** allows the system to continuously learn and adjust its behavior based on new experiences in real-time. This requires self-supervision as

well as data efficiency and representations with powerful generalizability that lightweight algorithms can utilize.

1.4 Contributions

The work in this thesis describes novel improvements in the perception component of an autonomous off-road driving software stack, deployed on a Yamaha Viking All-Terrain Vehicle (ATV). The key philosophy behind this work is to not only allow any robot to learn about off-road terrain purely through its prior experience, but to do so in a way that is efficient and intelligent enough that it can adapt to new terrain and experiences in real-time. To accomplish this, we present two major efforts that together aim to provide this functionality:

- A multi-modal **off-road driving dataset collected for the purposes of self-supervision**, released alongside a suite of tools that lower the barrier-to-entry for off-road driving research and encourage unified data collection efforts (adapted from [12]).
- A framework for an **online adaptative perception** system for off-road autonomy. We achieve self-supervision by leveraging proprioceptive cues such as IMU measurements and shock travel sensors, efficient adaptation via powerful feature representations, and uncertainty awareness to reason about unknown terrain and objects.

Chapter 2

Prior Work

2.1 Datasets

Collecting general driving datasets is a common pursuit, with the largest ones coming from companies gathering data in urban scenarios. For example, nuScenes by Motional is a dataset containing 1.4 million images and 390,000 LiDAR sweeps, with corresponding 3D object annotations [13, 14]. The Waymo Open Dataset is another example with over a million images and LiDAR pointclouds [15, 16]. Both datasets include infrastructure and tools to query and process these massive amounts of data in ways that support various downstream learning tasks and benchmarks that are not often seen in smaller datasets. There are also other efforts in collecting data for on-road environments, such as the RACECAR dataset that includes raw and processed sensor data from autonomous racecars driving at high speeds [17].

A number of earlier works have tried to circumvent the issues due to the lack of real-world off-road driving data by using simulation environments. Tremblay et al. show how training a neural network on multiple modalities in simulation can allow a robot to predict its dynamics in unseen real-world data [18]. Sivaprakasam et al. collect data in simulation of a robot driving over obstacles in order to train a model that predicts the difference between a desired path and its resulting path [19].

Now, more off-road datasets have been collected. RUGD contains over 7,000 images in a variety of off-road terrain with manually-labeled semantic segmentation masks [20]. Rellis-3D includes annotations for 6,235 images and 13,556 scans from

two different LiDARs, as well as the bags that they originally recorded which also contain IMU, GPS, and stereo image data. Sharma et al. created an off-road image dataset where, rather than creating semantic masks, they label regions in an image based on their traversability by different types of vehicles [21]. The first version of TartanDrive has also been available for two years and has been used for a number of tasks even outside the domain of off-road driving. Shah et al. have used this data as part of a bigger dataset that was used to train a large foundation model designed with vision-based robotic navigation in mind [22].

2.2 Self-Supervised Cost Learning and Online Adaptation

There exists a large number of existing works on learning self-supervised costmaps, both on and off-road. Some approaches leverage privileged information to supervise neural networks that predict map information at a given timestep [4, 11, 23, 24], but this information consists of semantic segmentation or come from handcrafted cost functions, both of which require hand labels and tuning. Some methods aim to circumvent this explicit labeling requirement by instead using expert demonstration data [1, 25], and while the supervision comes from the data collection process itself new challenges arise with ensuring demonstration quality. Recent works, taking inspiration from older methods ([9, 26, 27]), have explored the potential for proprioception as supervision as it allows for a strong robot-specific relationship between experience and cost. Some of these methods leverage signals such as residuals between planned and expected trajectories [3, 19, 28]. Others leverage IMU data to compute a roughness score which is in turn used as a heuristic for roughness [2, 29, 30, 31]. The primary challenge with these methods is producing a clean supervision signal, and acquiring large amounts of training data. The latter has recently been circumvented by leveraging pre-trained models and/or visual foundation models (VFMs), through which training data can be densified by associating unlabeled with labeled inputs [28, 32]. As a way of compensating for imperfect supervision, some works incorporate conditional value at risk (CVaR) allowing for the user to set their own risk tolerance. This CVaR value is computed by either predicting it directly, or predicting a distribution of costs

from which CVaR can then be predicted [1, 33].

These cost representations must also be adapted in real-time in order to learn about novel environments without forgetting prior experience. Some methods leverage structure present in incoming data in order to reason about which training samples to keep and which to discard [34, 35]. Chen and Ho et al. collect an ensemble of models and selectively train individual models based on their similarity to the current sample [24]. Seo et al. incorporate meta-learning into their training process such that their model can rapidly adapt to unseen data [31]. Mattamala et al. maintain a dual graph structure to represent prior experiences that are spread out spatially [28]. Note that the methods that use deep neural networks ([24, 28, 31]) are iteratively trained online, meaning that the amount of training that can be done in a given amount of time (and the speed at which it can learn) is dependent on the computational capability of the system.

Chapter 3

Experiment Setup and Data Collection

3.1 Vehicle

For our data collection and autonomy experiments, we use a Yamaha Viking All-Terrain Vehicle (ATV) (Fig. 3.1), first modified by Mai et al. [36]. We have modified it again in order to equip it with three LiDAR sensors. There are two Velodyne VLP-32 sensors mounted to the front of the roof of the vehicle, with one tilted downwards to increase coverage of the ground closer to the vehicle. There is also a Livox Mid-70, mounted under the MultiSense camera, which provides more information on objects directly in front of the vehicle.

3.2 Testing Sites

The site for collecting data is the same location in western Pennsylvania as TartanDrive 1.0: consisting of terrain such as narrow paths, dense foliage, rocky terrain, dirt paths, and steep hills over approximately 255 acres. It is worth noting that some areas of the site change significantly due to natural causes such as erosion or overgrowth, as well as expected seasonal changes as shown in Fig. 3.2. All the data included was acquired by a human tele-operating the vehicle.

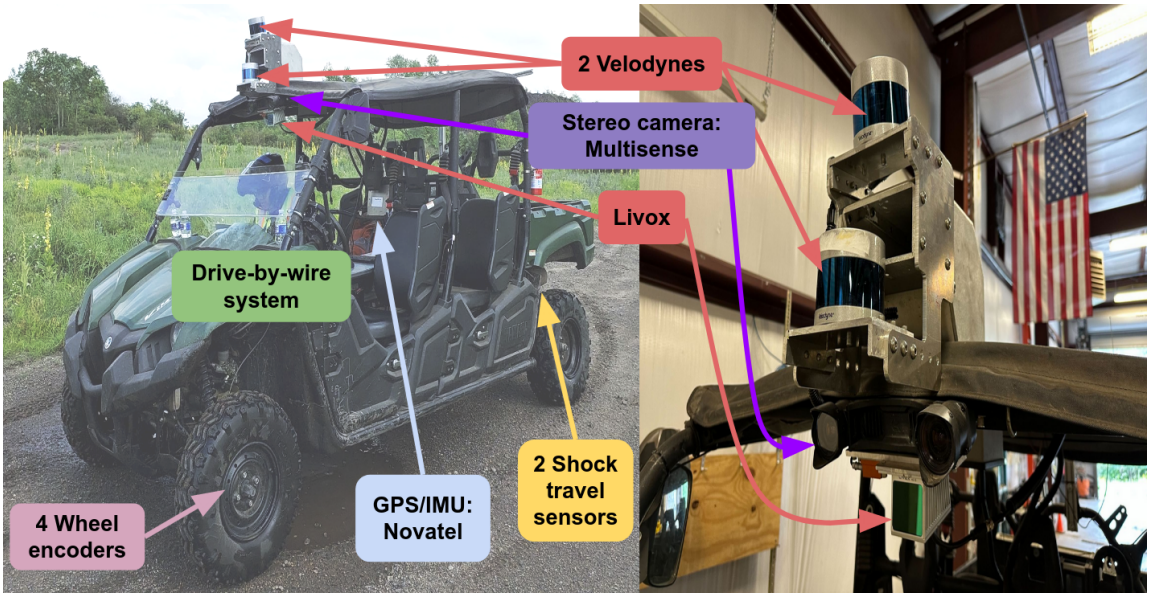

Figure 3.1: The ATV used for data collection (left); The primary sensor payload on the vehicle (right)

Figure 3.2: The data collection site has changed significantly over time. For example, some older dirt paths (top, shown in red) are now covered in tall grass (bottom).

3.3 Base Software Stack

As the majority of this work focuses on perception, we keep the other modules constant for all experiments:

3.3.1 State Estimation

We use Super Odometry [37] for state estimation, which leverages pointclouds from a Velodyne sensor and IMU data to provide odometry at 100Hz.

3.3.2 Planning

For long-horizon planning, we leverage prior information about the test site in the form of pre-defined courses. These courses consist of waypoints (ranging from 20-200 meters apart depending on the experiment), and are fed sequentially to the control module as the vehicle progresses.

3.3.3 Control

We use a model predicte path integral (MPPI) controller developed in [38]. We refer the reader to this work for more details but, at a high-level, sequences of actions are sampled multiple times from a distribution to generate trajectories, and a cost function is used to determine the best candidate trajectory. The cost function is determined by a weighted combination of both progress to the next waypoint as well as the cost calculated by rolling out the actions through a dynamics model and accumulating the costs of the rollout projected onto a 2D costmap provided by perception. The implementation has been modified by the author to also account for speedmaps, where if an action at exceeds the speed specified at its corresponding location in the speedmap, its corresponding trajectory will be ignored.

Chapter 4

TartanDrive 1 and 2 Datasets

4.1 Introduction

As the state-of-the-art in autonomous driving improves, robots are expected to handle increasingly complex tasks. In off-road driving, this translates to situations such as a robot knowing whether the dark brown path in front of it is dry dirt or thick mud, whether it is worth it to take a shortcut through tall grass, and when to preemptively increase velocity in order to make it up a hill. Navigating these decisions requires behaviors that are extremely difficult to robustly design and tune by hand. Many of the currently-researched solutions to this challenge involve the adoption of large neural networks and data-driven models. For the task of on-road driving in urban scenarios, there are several large datasets available [13, 14, 15, 16]. These datasets were feasible not only because of the resources present at the institutions that collected them, but also because of the inherent everyday nature of on-road driving. There exist fewer datasets for off-road driving, primarily because the nature of off-road terrain makes gathering enough data uniquely difficult. Collecting samples in simulation is often insufficient due to the complexity of calculating the dynamics of terrain in complex environments. In fact, most real-world scenarios that are hard to accurately simulate are the same ones which produce complex situations for autonomous off-road agents, such as dense foliage and slippery surfaces. However, collecting data in real-life presents logistics challenges such as preserving driver safety and dealing with frequent vehicle damage and wear. These difficulties have resulted in a scarcity of available

data for off-road driving.

While the number of off-road datasets has been growing [7, 20, 21, 39, 40, 41, 42, 43], many of them are still limited in sample size, modality, or difficulty [44]. This is often due to their focus on specific tasks such as semantic segmentation, which requires manual labeling effort. Scaling up these datasets would come either at the cost of time or label quality. It is possible to use multiple datasets together to train a model, but the variability of off-road terrain often causes inconsistency in labels across datasets [45]. The difficulty of generating large amounts of manually labeled data in off-road terrain suggests that self-supervised learning is essential to outperforming prior methods. Many works have shown that self-supervised methods can be used to learn strong representations [46, 47, 48], and some works have already shown how it can be used for various tasks in off-road driving environments [1, 2, 4, 24]. We argue that in order to scale up the amount of data for off-road tasks, datasets should be designed to be task-agnostic and with self-supervised learning in mind.

In 2021, we released TartanDrive, a multi-modal off-road driving dataset designed for dynamics modeling [39]. In this work we present TartanDrive 2.0, a larger dataset geared towards self-supervised learning methods. The contributions of this dataset over the previous generation are as follows:

1. **More sensors and data:** In our original work, we argue that multiple modalities are essential for learning models for off-road terrain. To that end, we add LiDAR as an additional modality. We also collect seven hours over the five hours of data present in the original dataset, at higher average speeds and including some new areas, as well as areas present in the previous dataset that have dramatically changed over the past two years.
2. **Better infrastructure for end-users:** We have improved the infrastructure in order to improve ease-of-use and utility of the dataset, such as the ability to re-process the data based on user-specified configurations. We also implement a metadata system that filters and groups data by a variety of properties.
3. **Open-source data collection tooling:** Our data collection process has been constructed in a way that allows us to continually release more data over time as the seasons change and as we add more modalities. By releasing our tools and framework, others can collect their data and easily merge it with our own

to create larger datasets.

4.2 Data Collection

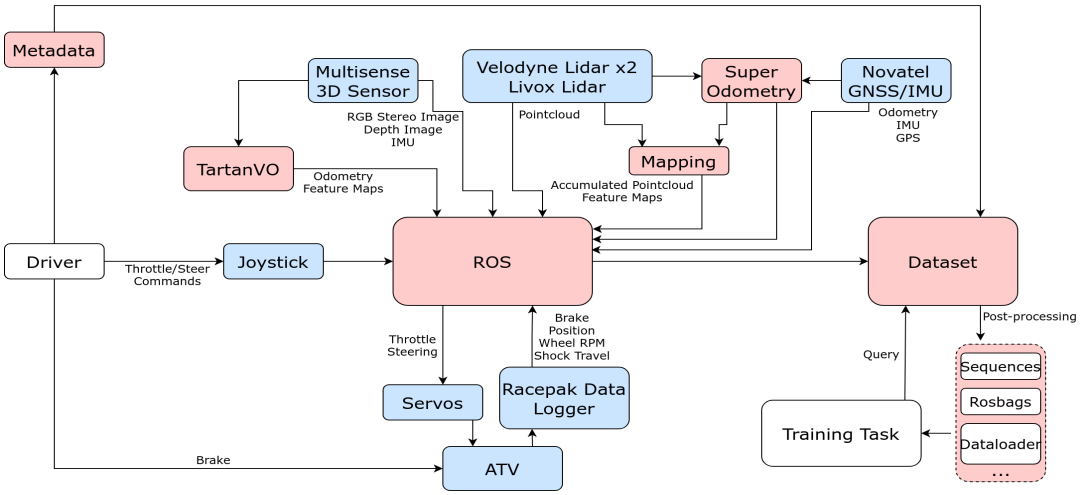

Figure 4.1: The high-level flow of information in our data collection process, with software in red and hardware in blue.

All the data included was acquired by a human tele-operating the vehicle, with the overall flow of data outlined in Fig. 5.1. Each sequence of data is annotated with the following metadata:

- Driver ID and robot
- Number of people in the vehicle
- Date/time
- Context (e.g. data collection)
- Weather conditions (e.g. dry, damp, snow)
- Lighting conditions (e.g. sunny, overcast, sunset)
- Course ID or general location (from a pre-defined list)

as well as any other information relevant to that specific run. We emphasize that recording this metadata makes the dataset significantly easier to use, especially as the dataset becomes larger. At the end of the collection, a post-processor records

information such as top speed, average speed, duration, and sensors present. During data collection, a co-pilot takes time-stamped annotations of relevant or unique events such as sensor failure or something uncommon such as a deer spotting in front of the vehicle. Additionally, in some runs the co-pilot periodically annotates a weak driving score ranging from 1-5 where 5 signifies ideal driving. Recording this metadata provides more structure in the data which significantly improves the utility of the dataset when used in combination with our post-processing pipelines.

4.3 Raw Data

Most of the sensors on our platform have already been detailed [39, 49], but all raw data is briefly summarized below:

4.3.1 Pointclouds

We record incoming pointclouds from two Velodyne VLP-32 LiDAR sensors and a Livox Mid-70. We also provide extrinsics so that they can be merged into a single pointcloud. An example of the coverage they provide is shown in Fig. 4.2. All LiDARs are configured to run at 10Hz.

4.3.2 Images

A Carnegie Robotics MultiSense S21 is used to provide stereo images, specifically greyscale images from both cameras as well as RGB images from the left camera at 10Hz each.

4.3.3 IMU and Pose

A NovAtel PROPAK-V3-RT2i GNSS provides IMU data at 100Hz. This is fused with incoming GPS data to provide a pose estimate at 50Hz. The MultiSense also provides IMU data at 400Hz.

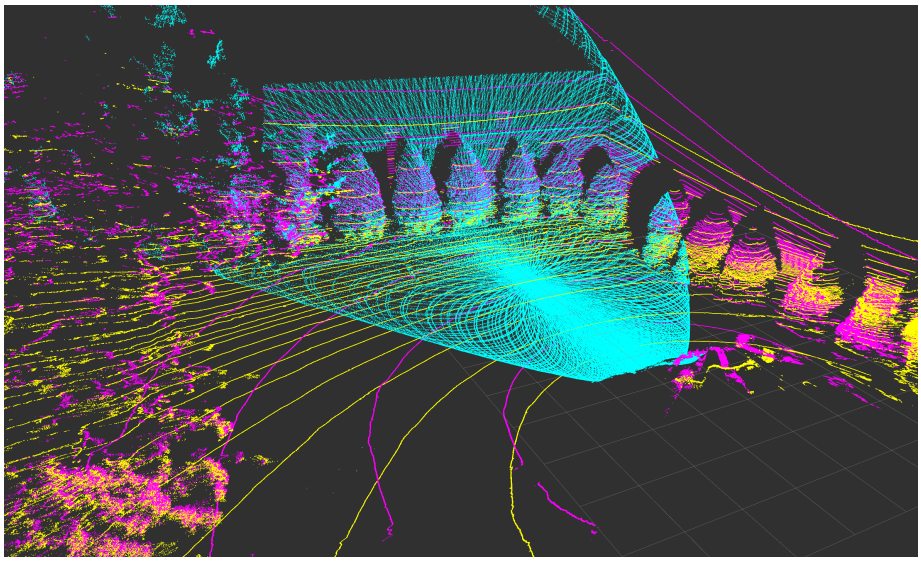

Figure 4.2: Example coverage provided by our LiDAR sensors. Purple and yellow points come from the two Velodynes, and cyan points from the Livox. Note that the scans shown are aggregated over time to better demonstrate the full coverage of the Livox scan pattern.

4.3.4 Teleoperation

The driver uses a joystick controller to send steering commands, which are then recorded alongside the values of the current steering angle (actuation on steering commands is not instantaneous). A Racepak G2X Pro Data Logger is used to provide the positions of the acceleration and brake pedals.

4.3.5 Proprioceptive Information

The Racepak is also used to record RPMs for each wheel and suspension shock travel for the rear two wheels. Shock travel data for the front two wheels is omitted due to frequent damage to the sensors during aggressive driving maneuvers, but will be included in future data releases.

4.4 Post-Processed Data

In order to increase utility and ease-of-use of the dataset, we provide the raw data and the outputs from some existing modules that have been integrated into our software stack (Fig. 5.1) and commonly use as inputs to our own algorithms:

4.4.1 TartanVO

We run TartanVO [50] on the platform, which takes in the stereo images from the MultiSense as input. It outputs an odometry estimate, a predicted pointcloud, and top-down height and RGB maps, all at 10Hz.

4.4.2 Roughness Cost

We produce a roughness cost that describes the bumpiness of terrain as we drive over it, derived from a sliding window of Z-axis linear acceleration values from the IMU as described in Guaman Castro et al. [2].

4.4.3 Odometry and Registered Pointcloud

In addition to the odometry provided by the Novatel system, we also provide an estimated output by Super Odometry [37]. The high accuracy and frequency of this output allows for cleaner results in tasks such as pointcloud registration which is in turn important for other downstream tasks. It also serves as a baseline for others to benchmark their own odometry methods.

4.4.4 Local LiDAR Maps

Using the registered pointcloud provided by Super Odometry, we generate a birds-eye-view feature map 200x200m wide at .5m resolution. This map provides geometric information about the environment (Fig. 4.3), with features consisting of the following:

- Min/Max/Mean Height of Points
- Roughness
- SVD Features

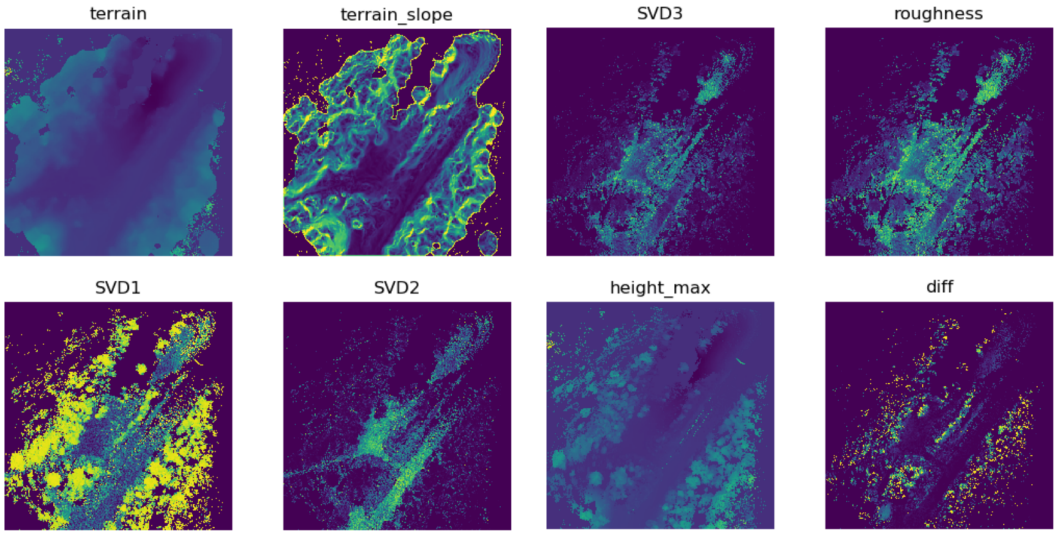

Figure 4.3: Using the registered pointcloud, we provide local maps that contain various geometric features.

- Estimated Ground Height
- Estimated Ground Slope (X, Y, Magnitude)

4.5 Data Pipelines

4.5.1 Formatting

We provide the data in two main formats, the first one being the rosbags that they were originally recorded as. This allows users to test how their algorithms might perform in different environments in real-time. The second format is as a set of sequences similar to the KITTI format [51]. For each bag, a folder is created and a subfolder for each modality is initialized. The samples across all modalities are then timesynced and then placed in their respective subfolders (e.g. 'pointcloud_1/0000.bin' and 'image/0000.jpg' are associated with each other).

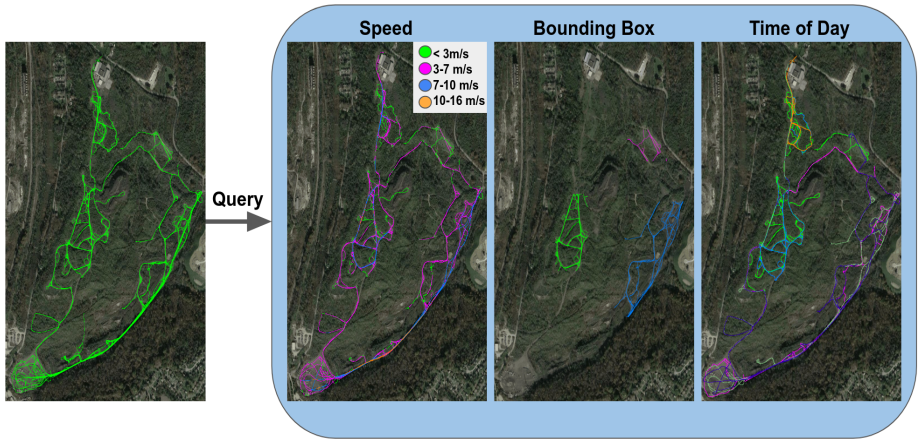

Figure 4.4: Our dataset covers over 255 acres collected over seven hours. Using the accompanying metadata, it can easily be split into groups by aspects such as speed, GPS bounding box, and time of day.

4.5.2 Reconfiguration

We have post-processed our data in a way that makes sense for our own algorithms, but our ATV platform is somewhat unique. Across different robots, the optimal parameters for elements such as sample frequency, map size, and map resolution vary. For example, a smaller robot with better agility might require a map with finer than .5m resolution. In order to make our data more directly applicable to other platforms, we provide scripts that allow users to regenerate the dataset with different parameters as they see fit.

4.5.3 Utilities

As previously mentioned, we collect metadata and some annotations during data collection. We provide scripts to take advantage of this metadata in order to filter the data by various elements and therefore increase the utility of the dataset for different learning tasks. For example, given the whole dataset, a user can easily create subsets grouped by components such as speed, GPS bounding box (we provide a GUI for easily generating bounding boxes), lighting conditions, or driver (Fig. 4.4). This is especially useful for testing model generalizability by training on one location/condition, and testing on another.

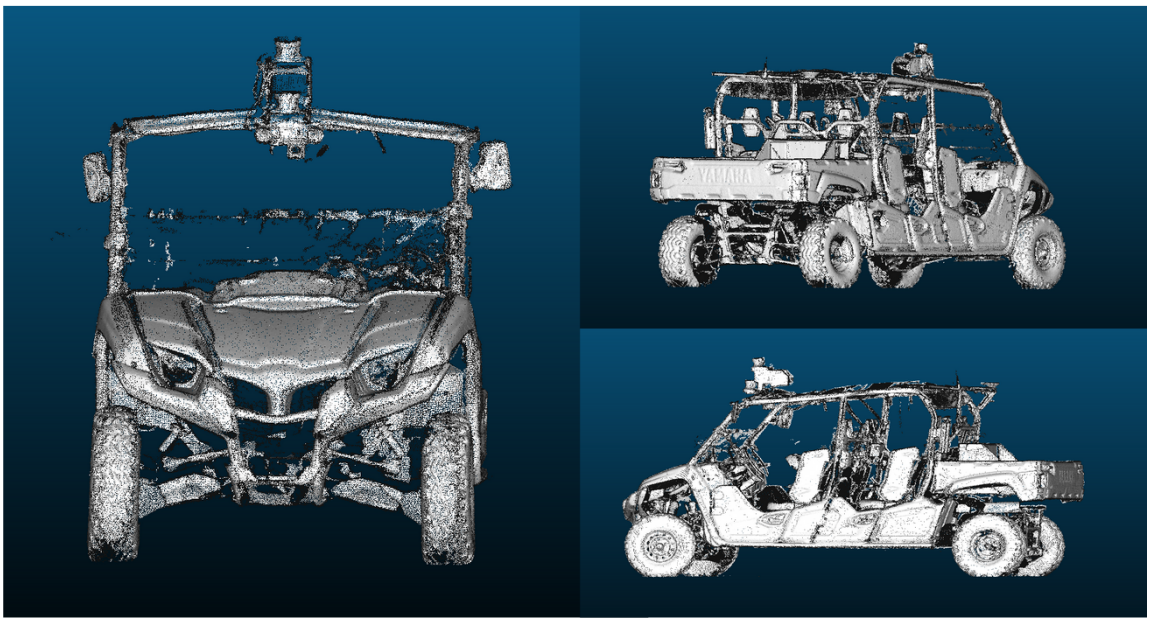

Figure 4.5: The 3D scan of the ATV. The rear half of the scan is cropped in the left picture to increase clarity of the front payload in the image.

Using a Faro Focus Scanner, we also generated a 3D pointcloud model of the ATV as shown in Fig. 4.5 and provide it with the dataset so that users can take their own measurements (e.g. distance between the wheel axle and some arbitrary point on the vehicle, or distance between the GPS antenna and the ground) for their own specific experiments.

4.5.4 Common Framework

The infrastructure we have developed has streamlined our data collection procedures to make it easy to train models on processed datasets as well as test our algorithms in real-time. By continuing to follow the same procedures, the data we collect in the future can easily be merged into our existing datasets. Likewise, other researchers can use our tools when collecting their own off-road driving data which could eventually lead to a larger multi-site, and multi-vehicle dataset.

Dataset	State	Action	Image	Pointcloud	Heightmap	RGBmap
RUGD [20]	No	No	Yes	No	No	No
Rellis 3D [40]	Yes	Yes	Yes	Yes	No	No
Wild-Places [42]	Yes	No	No	Yes	No	No
Montmorency [52]	Yes	Yes	Yes	Yes	Yes	No
Verti-Wheelers [41]	Yes	Yes	Yes	No	No	No
TartanDrive 1.0 [39]	Yes	Yes	Yes	No	Yes	Yes
TartanDrive 2.0 (Ours)	Yes	Yes	Yes	Yes	Yes	Yes

Table 4.1: (Pt. 1)Overview and comparison of various off-road driving datasets

Dataset	IMU	Wheel RPM	Shocks	Metadata	Labels
RUGD [20]	No	No	No	No	Yes
Rellis 3D [40]	Yes	No	No	No	Yes
Wild-Places [42]	No	No	No	No	Yes
Montmorency [52]	Yes	No	No	No	Yes
Verti-Wheelers [41]	Yes	Yes	No	No	No
TartanDrive 1.0 [39]	Yes	Yes	Yes	No	No
TartanDrive 2.0 (Ours)	Yes	Yes	Yes	Yes	No

Table 4.2: (Pt. 2)Overview and comparison of various off-road driving datasets

4.6 Impact on Current and Future Research

While we do not provide any explicit supervision labels (e.g. segmentation masks, classification labels) outside of annotations, the several modalities and actions that we provide facilitate a number of self-supervised learning tasks. Our original TartanDrive has already benefited off-road research immensely (Table 4.3), and we believe the categories listed in Tables 4.1, 4.2 are only a subset of the research topics TartanDrive 2.0 can bolster.

4.6.1 Perception - Cross-Modal Supervision:

Our data was used by Guaman et al. to visually predict a bumpiness cost supervised by an IMU-derived metric [2]. With the addition of LiDAR in our dataset, we can advance algorithms by providing information in scenarios where cameras fail (Fig. 4.6) and learning from a much more accurate source of odometry and depth (Fig. 4.7). For example, Chen et al. accumulates near-range LiDAR measurements to

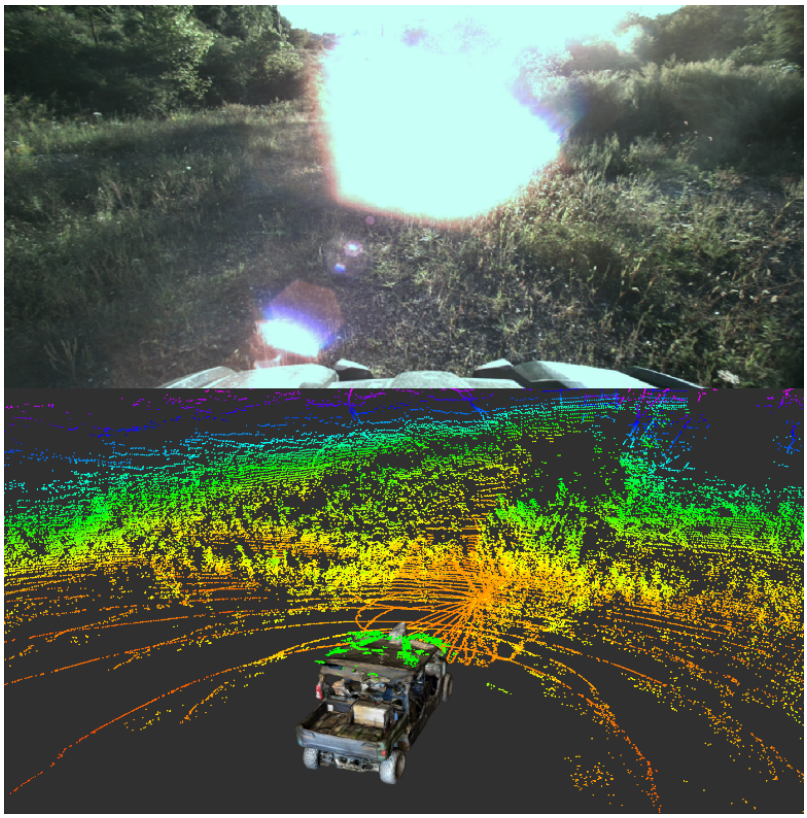

Figure 4.6: In cases where conditions like poor lighting affect camera performance, LiDARs can allow a robot to still understand its surroundings.

learn long-range visual traversability model [24]. Meng et al. train a model that uses images to predict feature maps supervised by lidar inputs [4]. However, these works rely on off-road driving datasets that are not publicly available. By releasing data in a similar domain with the same modalities, we lower the barrier to entry for expanding on this field of research, and allow a common point of comparison.

4.6.2 Perception - Map Completion:

Prediction of occluded and sparsely sensed areas enables faster safe navigation in occluded area [4, 53]. Our generated accumulated maps are being used as a supervisory signal in our ongoing research in map completion and can also serve as a common benchmark for other approaches.

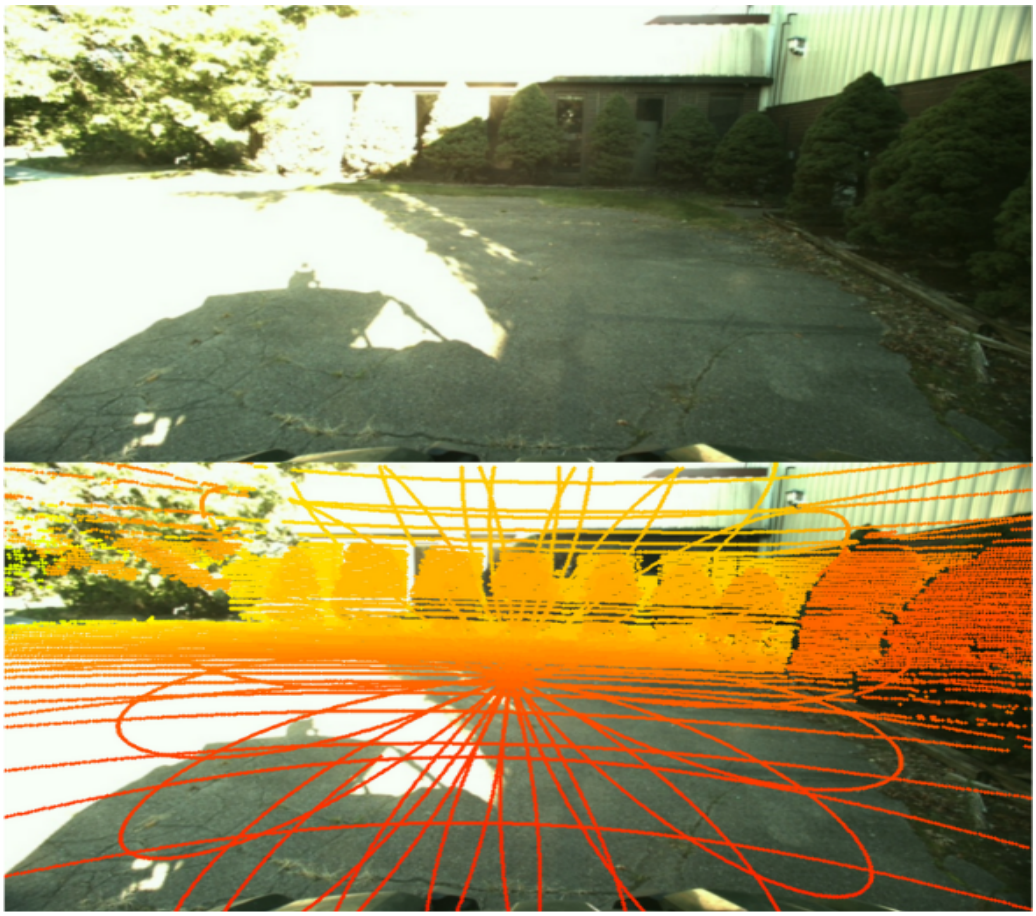

Figure 4.7: An example of the overlap between a single scan from all 3 LiDARS and our camera. Points are colored by forward distance from the vehicle.

4.6.3 Perception - Ground Height Estimation:

An understanding of the support ground surface in off-road terrain is essential for successful navigation. We enable methods of learning this feature by providing multiple modalities as inputs and a 3D model of the car that can be used alongside odometry estimates to provide supervision of the true ground surface.

4.6.4 Planning - Learning from Demonstration:

The actions derived from human teleoperation coupled with sensor inputs can be used to learn models that can reason about what areas are more traversable than others. Triest et al. treats teleoperated driving as expert demonstrations and use

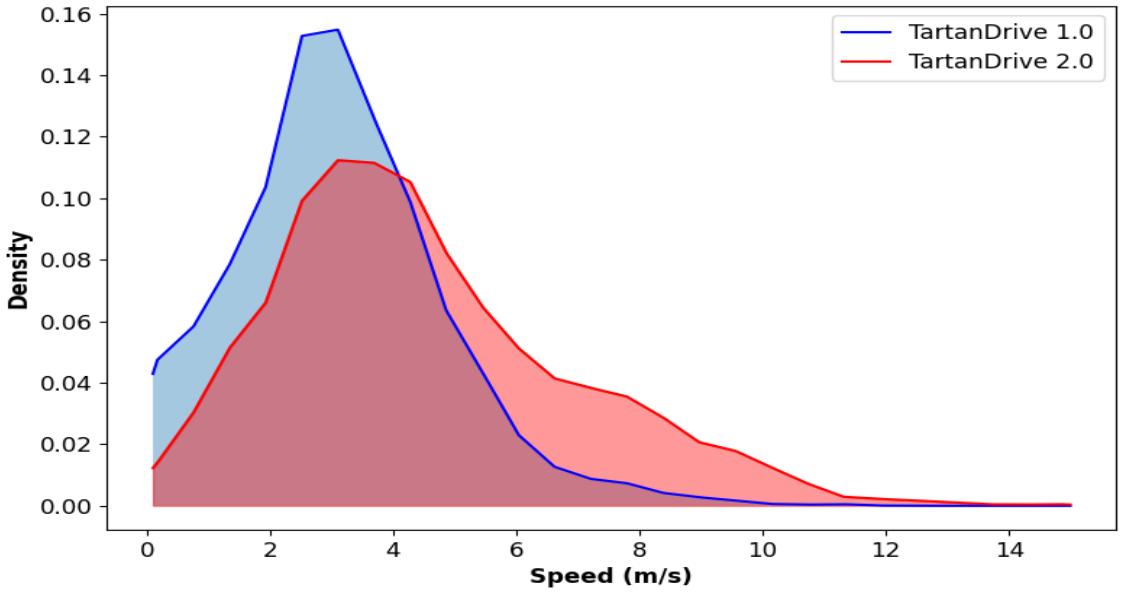

Figure 4.8: In TartanDrive 2.0 we collect a higher concentration of data at higher speeds than in TartanDrive 1.0, reaching up to 15m/s.

inverse reinforcement learning to birds-eye-view costmaps from the LiDAR feature maps [1].

4.6.5 Controls - Aggressive Maneuver Driving:

The unstructured nature of complex terrain makes it difficult to drive aggressively at high speeds without losing control. Our new dataset contains a higher proportion of speeds beyond 7m/s than before (Fig. 4.8). Some of our ongoing research on learning vehicle dynamics models is supported by this data.

Table 4.3: Works that Utilize TartanDrive 1 and 2

Method	**Task**
Learning Risk-Aware Costmaps... [1]	Perception
How Does it Feel? [2]	Perception
Terrain Depth Estimation ... [54]	Perception
Deep Bayesian Future Fusion [55]	Perception
PIAug [56]	Dynamics Prediction
PhysORD [57]	Dynamics Prediction
Cross-Embodiment Learning... [58]	Policy Learning
General Navigation Model [59]	Policy Learning
Foundation Model for Visual Navigation [22]	Policy Learning

Chapter 5

Online-Adaptive Risk-Aware Costmaps and Speedmaps with Uncertainty Detection

5.1 Motivation

Utilizing sensor data in the same format as provided in TartanDrive 2.0, we aim to develop a perception system that can predict expressive information about its environment. In this work we propose a method for learning robot-specific relationships between terrain, speed, and roughness in real time. We first introduce a modular mapping pipeline to project information from the camera into a top-down map space in the same form of our desired costmaps (this also allows us to determine what visual features the tires traverse over). We then predict a costmap that leverages the robots experience to predict the roughness of its surroundings as well as a speedmap that informs the downstream controller how fast to drive without exceeding a user-defined roughness threshold. Both of these maps adapt in real-time. In order to distinguish unknown terrain that the robot can learn about from uncertain obstacles that could be lethal, we use the visual features to produce an uncertainty estimate.

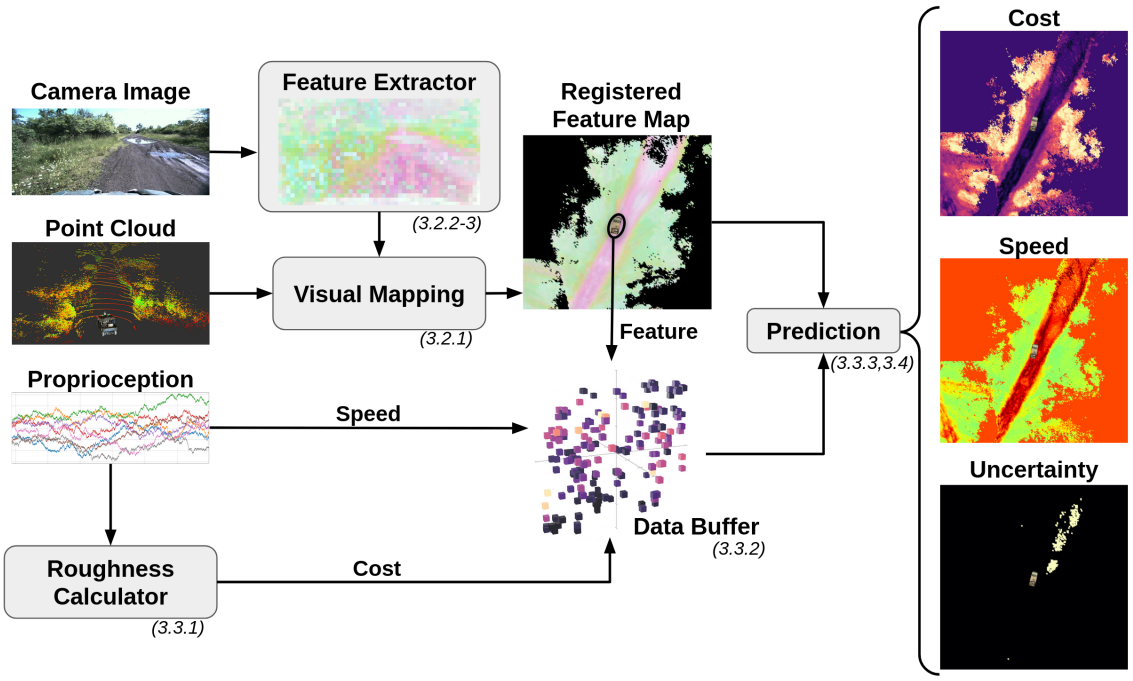

Figure 5.1: We introduce a method for predicting costmaps, speedmaps, and uncertainty that leverages visual foundation models and proprioceptive self-supervision to adapt online.

5.2 Perception Backend: Modular Visual BEV Mapping Framework

Visual features from models such as semantic segmentation networks and visual foundation models have been shown to be useful in a variety of domains. To explore their efficacy for our task, we design a modular system that projects these visual features from the image space to a birds-eye-view (BEV) representation, where they are aggregated into a map.

5.2.1 Feature Mapping

Pixel-level features in the image frame can be computed via feature extractor f_θ, providing the mapping:

$$f_\theta(I_{3xHxW}) = D_{CxHxW} \tag{5.1}$$

where I is an RGB image, D is a "featurized" image, and C is the number of features (C=3 in the case where the RGB image is passed through instead of extracting features).

Given these features, we leverage standard camera geometry techniques to project the pointcloud from a lidar sensor into the camera image with intrinsic matrix K and extrinsic rotation and translation R, t between the two sensors:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \begin{pmatrix} R|t \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{5.2}$$

to obtain the corresponding image coordinates u, v for each point $P = (x, y, z)^T$, in turn allowing us to associate each 3D point with a visual feature. These featurized points are then projected into a BEV map, where the value of a cell in the map is equal to the average feature of all points that correspond to that cell. Since the lidar scans are relatively sparse a single map frame will also be sparse, so we first align the map at time $t-1$ with the map at time t using odometry, and then aggregate them using an exponential moving average such that for cell m in map M:

$$m_t = \alpha m_t + (1 - \alpha) m_{t-1} \tag{5.3}$$

This process is outlined at a high-level in 5.1 with more detail in Fig. 5.2.

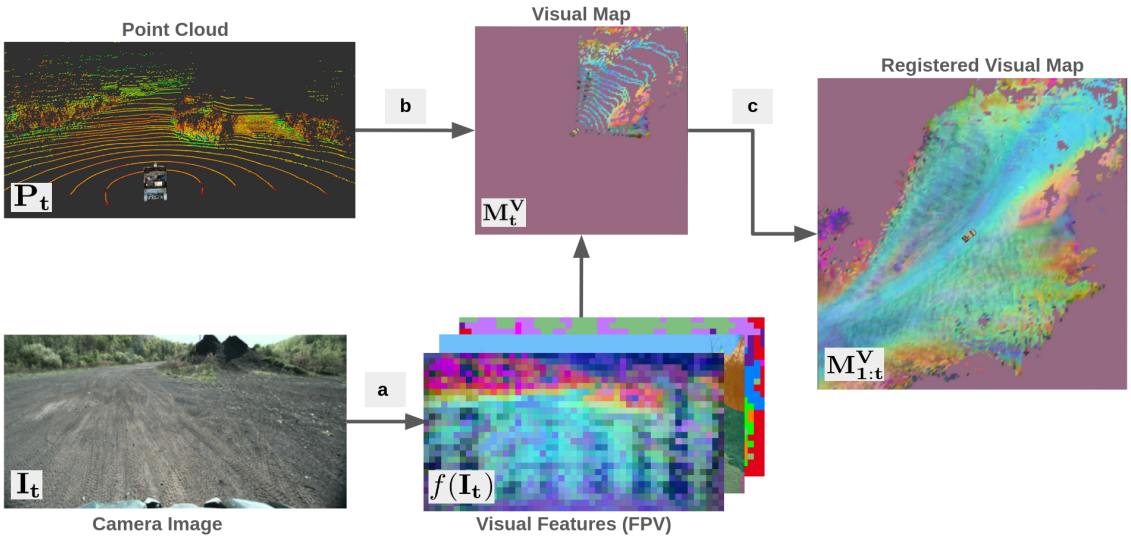

Figure 5.2: The overall flow of information from the raw sensor data into an visual feature map. (a): Features are extracted using a pretrained neural network and their dimensions are reduced to less than 16. (b): Pointclouds from the lidar sensor are used to project features from the image space into a BEV map space. (c): Single maps are registered and aggregated over time to form a visual map for downstream tasks.

5.2.2 Vision Modules

Utilizing the above framework, we integrate various feature extractors f to provide information for downstream tasks.

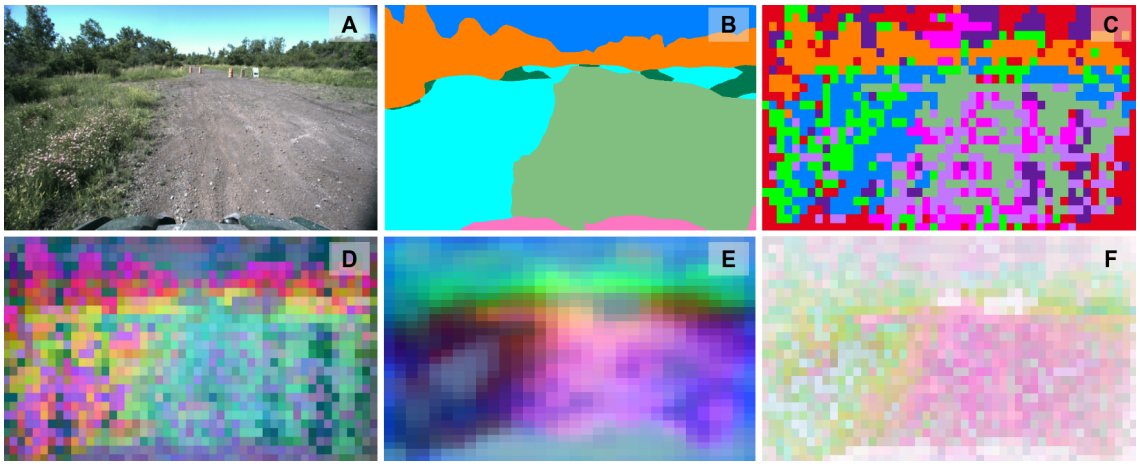

Figure 5.3: The various visual features available through our pipeline. (a): The original input image. (b): Output segmentation labels from GANav. (c): Hard cluster assignments using VLAD on DINOv2. (d): Visualization of the top 3 components of the PCA representation of DINOv2. (e): Visualization of the top 3 components of the PCA representation of AM-RADIO. (f): Visualization of 3 cluster residuals of the representation of DINOv2.

GANav

As an option for semantic segmentation, we leverage GANav [60] a state-of-the-art approach for off-road semantic segmentation, shown in Fig. 5.3b. It has been pre-trained using the Rellis-3D dataset [40] and finetuned on labelled subset of data at our test site.

DINOv2

Recently, a visual foundation model DINOv2 [47] has been released, with multiple versions available that have been shown to be useful. It has been integrated into our stack, using the approach from Keetha et al. ([61]) such that any layer from any version can be used to extract features for downstream tasks.

AM-RADIO

We also integrate AM-RADIO [62], a work that claims to have distilled features from CLIP [63], DINOv2, and Segment-Anything-Model (SAM) [64] into one model, for the purpose of evaluating it against the other models on our task.

5.2.3 Dimensionality Reduction

While the visual foundation models produce features with hundreds of channels per pixel, representing them all in a BEV representation at real-time can become intractable. Offline, we generate pixel-level embeddings on a subset of training data X. We then randomly sample C-dimensional embeddings from all generated feature images, on which we leverage one of two dimensionality reduction techniques in our experiments.

PCA

Principal Component Analysis (PCA) is a standard dimensionality reduction approach in which the singular value decomposition of the standardized features is first generated:

$$U\Sigma V^T = X_{std} \tag{5.4}$$

Then, the vectors from V corresponding to the top n singular values in Σ can be selected and used to obtain the reduced X_{pca}

$$X_{pca} = X_{std}V_n^T \tag{5.5}$$

The top three eigenvectors of both DINOv2 and AM-RADIO are shown in Fig. 5.3 d and e, respectively.

VLAD-inspired Descriptors

Vector of locally aggregated descriptors (VLAD) is a popular method for generating descriptors for the place-recognition task [65]. K-Means clustering is used to generate k feature clusters, and a descriptor is generated by comparing the features to each cluster and summing the residuals. The authors of AnyLoc [61] analyze the clusters generated when this technique is applied to the output of DINOv2 and observe that they have highly semantic properties. This inspired us to use it as a dimensionality reduction technique, where each feature is instead represented as its distance to the clusters F such that given the original feature vector X, the k-th element in reduced feature vector X_{VLAD} is computed by:

$$X_{VLAD}[k] = \|X - F_k\|_1 \tag{5.6}$$

This representation can be used to create a psuedo-semantic label for each pixel by taking its hard assignment to a cluster (Fig. 5.3c), or all the residuals can be used together as features for downstream perception (Fig. 5.3f).

5.3 Adapting Perception with Online Experience

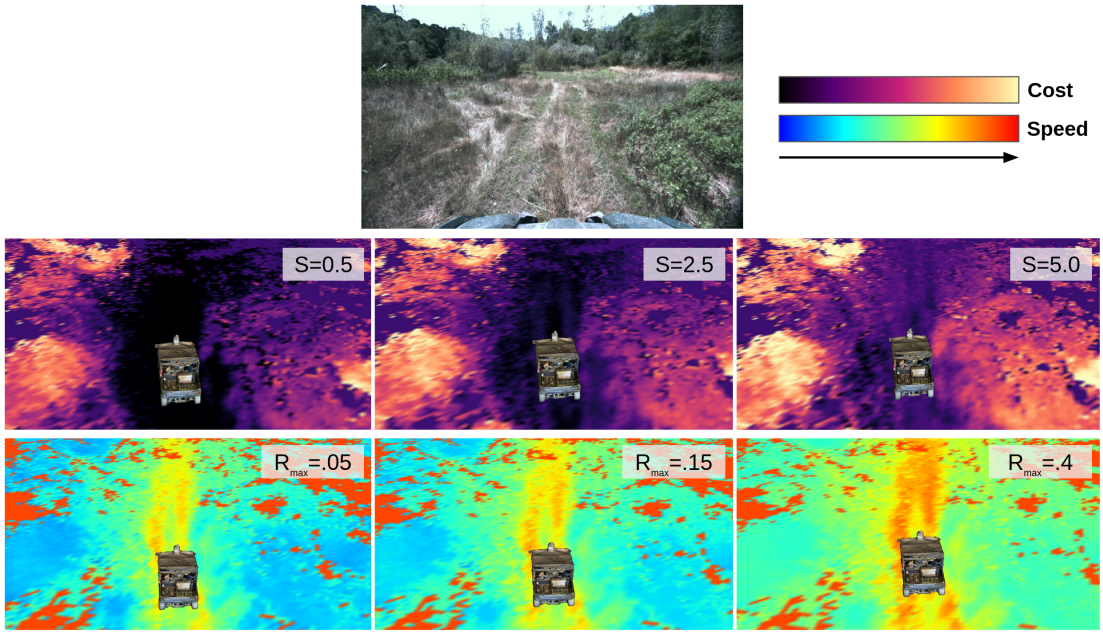

Figure 5.4: We introduce a framework for leveraging visual foundation model features and proprioceptive cues to predict costmaps and speedmaps for off-road navigation. The top row depicts costmaps, conditioned on speed increasing from left to right (note the increase in cost in the tall-grass areas with higher speed). The bottom row depicts speedmaps, with a user-set maximum roughness threshold increasing from left to right.

5.3.1 Curation of Self-Supervised Signal

For a robot to adapt from its own experience, it requires a signal that associates different types of terrains with different costs in a way that sufficiently matches

human intuition. In line with the work by Castro et al. ([2]), we also take inspiration from prior work ([27, 66, 67]) and use the bandpower of the robot's Z-axis linear acceleration provided by the IMU, where bandpower BP can be calculated through the following formula:

$$BP(s, f_{min}, f_{max}) = \int_{f_{min}}^{f_{max}} S(f)\ df \tag{5.7}$$

where S is the power spectral density (PSD) of a signal over the past s seconds, and f_{min}, f_{max} describe the frequency range used. We use the same methods to compute this as in [68], specifcally Welch's method [69] to compute PSD and Simpson's rule to evaluate the integral. While many works, including [2, 31] leverage Z-axis acceleration from IMU data alone, we find that the complexity of the vehicle used plays a significant part in the effectiveness of the bandpower signal in representing roughness accurately. To address this, we also compute the bandpower of the X and Y accelerations and of the shock travel sensor data.

In order to condense the multiple values computed into a single roughness score, we first collect a small driving dataset consisting of the following:

- Brief (20-40 second) traversals over three different types of terrain at low, medium, and high speeds
- One longer (10 minutes) trajectory in which a wide range of grass, trail, and bumpy terrains were traversed

where each trajectory was periodically annotated by a passenger in the vehicle with a traversability score in the range 0-1. We then define the following parameters we wish to optimize:

- f_{min}, f_{max} specific to each signal, $0 \leq f \leq 50$
- A weight specific to each signal, $0 \leq w \leq 1$
- Window size (seconds) of data to operate on, $0.5 \leq s \leq 2$

such that the un-normalized roughness R' is computed by:

$$R' = \sum_{i \in [a_x, a_y, a_z, shock...]} w_i BP(s_i, f_i^{min}, f_i^{max}) \tag{5.8}$$

The final costs are then normalized to a range 0-1 based on statistics computed

from the dataset. In order to obtain the best set of parameters we randomly sample a value for each parameter, compute the resulting roughness values, and compare them to the human annotations using cumulative error as a metric.

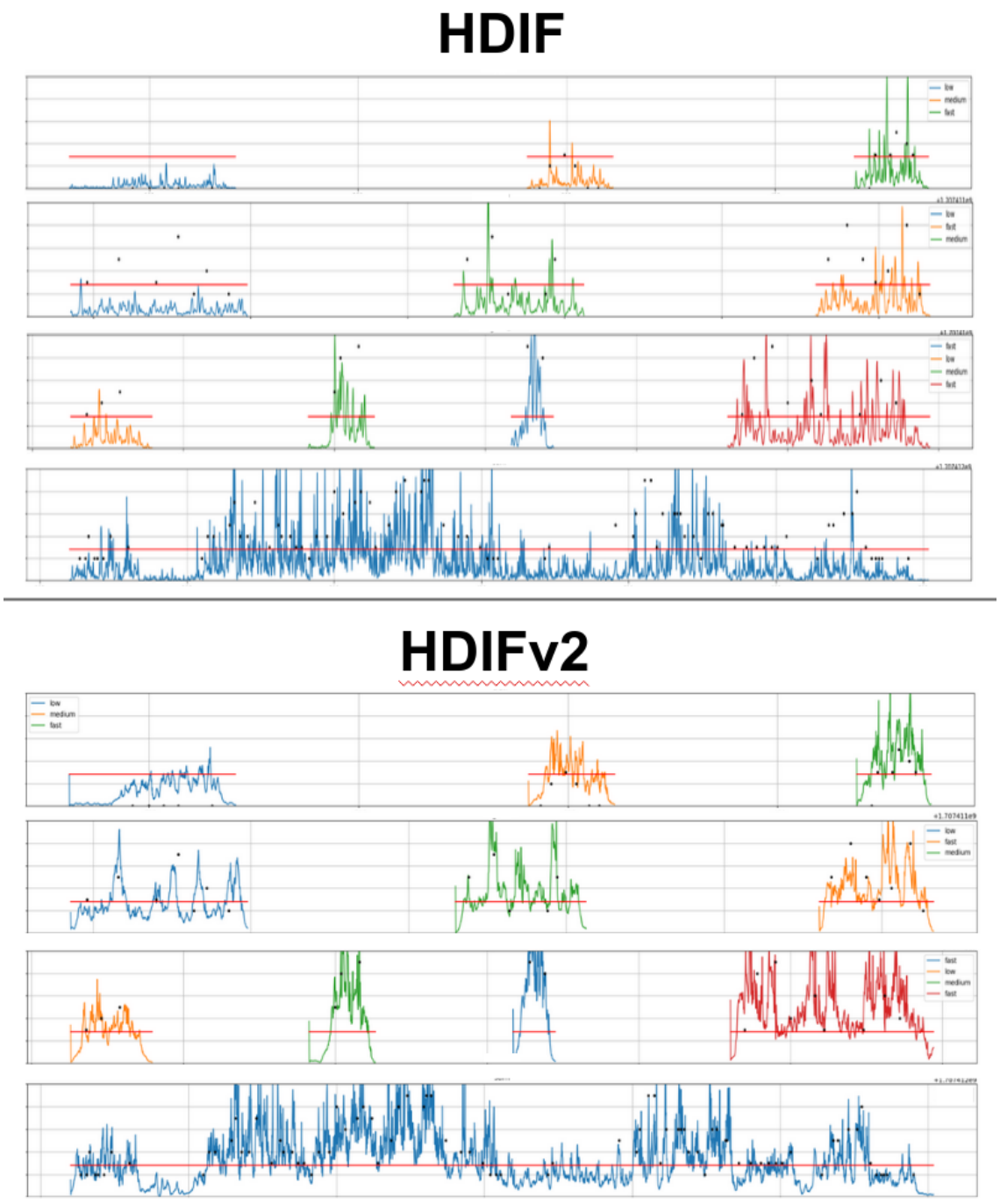

Figure 5.5: Comparison of the original roughness cost function from HDIF and our new version. The rows from top to bottom represent: driving on relatively flat trails, two different bumpy and grassy areas, and one long run covering a wide set of terrain at various speeds. For the top three graphs, orange represents slow (0-3m/s) speed, green represents medium (3-6m/s), and blue and red represent fast (6-10m/s). The black dots represent human annotations of experienced roughness. The red lines set a reference point to demonstrate that roughness increases roughly with speed.

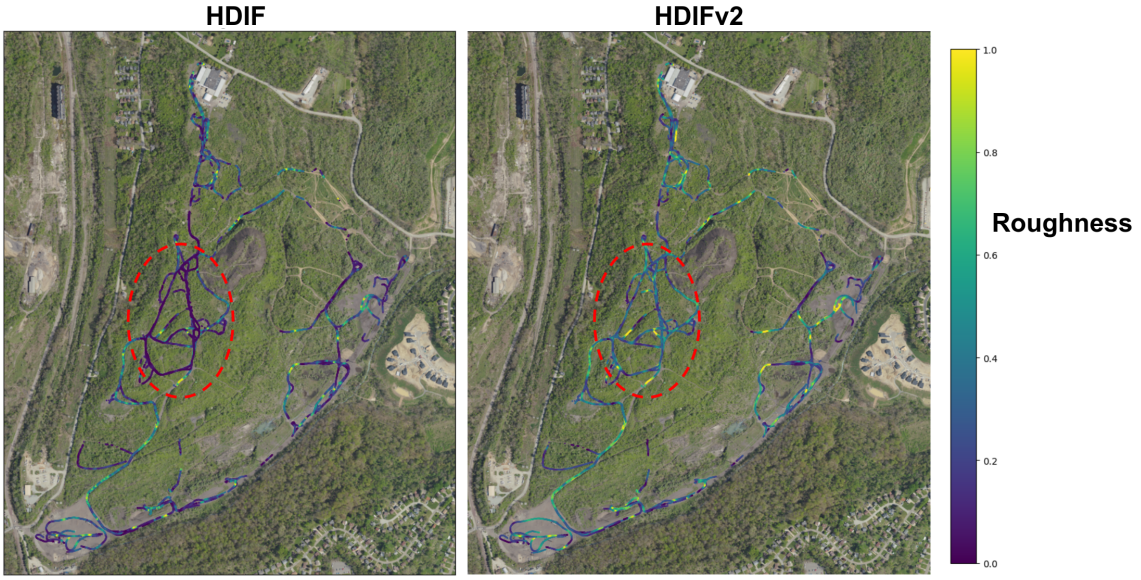

Figure 5.6: Comparison of the original roughness cost function and the new version. The new version better captures smaller variations in cost in similar types of terrain, as shown in the trail area circled in red.

5.3.2 Intelligent Data Maintenance

We leverage the visual BEV mapping described above as a perceptual representation and have the robot store experience as it drives. A buffer of samples is maintained and updated over time, where the sample collected at time t contains the following:

1. O_t - The observed visual feature obtained by taking the value of the BEV map corresponding to the location of a vehicle tire at time t
2. S_t - The speed that the vehicle was traveling
3. R_t - The roughness that the vehicle experienced

We use a fixed-size buffer due to memory and runtime constraints, which means we require a way to choose what data is removed in order to make room for new data. We could naively adopt a "first in, first out" strategy, but this would result in vulnerability to problems such as catastrophic forgetting, in which the system forgets old experiences in the process of learning new ones. For example if the robot drives on trail but then drives consistently in vegetation for several minutes, all the samples collected on trail would eventually be removed from the buffer. In order to prevent

Algorithm 1 Buffer Insertion Strategy

Input: Visual feature map M_f, current pose p, current speed S, current roughness R, data buffer B
if $v > 0$ and t % $update\_rate = 0$ **then**
  $O \leftarrow M_f[p]$
  $c = \arg\min(O)$
  if B is full **then**
    $c_{common} \leftarrow mode(\arg\min(B^O))$
    $B_{common} \leftarrow B[B^O = c_{common}]$
    $s_{common} \leftarrow mode(B^S_{common})$
    B.remove(random.choice($B^{S=s_{common}}_{common}$))
  end if
  B.append(O, S, R)
end if

this, we implement a strategy that is robust to this type of situation and ensures an even distribution of data across the feature space. The VLAD features in the BEV representation in by nature describe distance of observations to pre-defined clusters. If we assume each cluster maps to a semantic class (which we find to be true), then we can assume that a given observation is of a semantic class corresponding to the index of its smallest element. When the buffer is full, rather than throwing out the oldest sample, a random sample of the most common semantic class in the buffer can be removed. Intuitively, not only observation O but also vehicle speed S affect the roughness R experienced (demonstrated in Castro et al. [2]), so we take this strategy a step further by looking at all the most semantically-common samples and removing one with the most common speed. The psuedocode for this process is outlined in Alg. 1.

Table 5.1: Average Total Sum of Distances Between Points in Buffer

Strategy	Scenario 1	Scenario 2	Scenario 3
FIFO	3.55	3.60	3.59
Ours	**4.90**	**4.35**	**3.97**

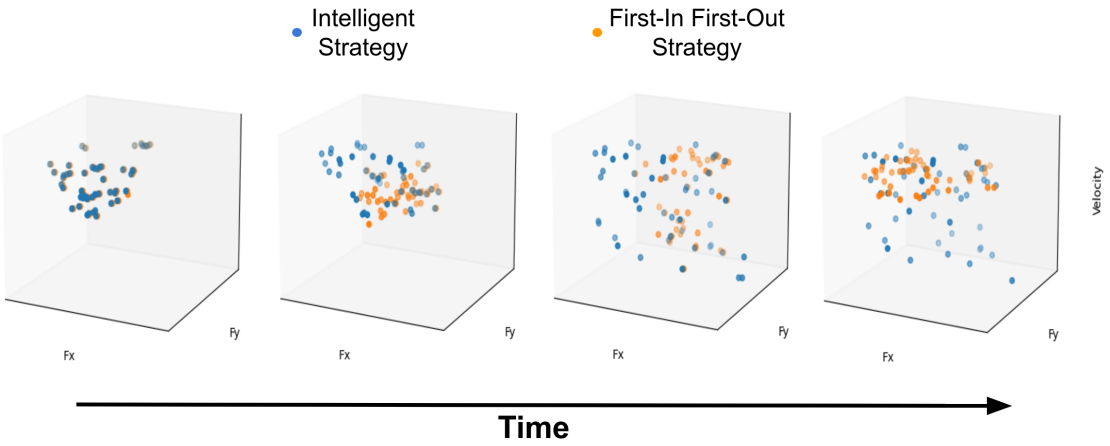

Figure 5.7: Example scenario of data available in the training buffer over time, using a naive first-in first-out (FIFO) strategy versus our strategy that leverages the inherent structure of the data to maintain a distribution that covers a diverse set of experience over time. Fx and Fy represent a t_SNE projection of the observed features, and the vertical axis is speed. Unlike our strategy, the FIFO strategy is susceptible to forgetting prior terrain and velocities over time in order to make room for new data.

To verify this strategy, we compute the average total sum of distances between all points in the buffer, shown in Table 5.1 where the higher values for our method indicate a better coverage of the sample space in all of three different scenarios.

5.3.3 Costmap and Speedmap Estimation

Given prior experience through which roughness is associated with visual features traversed at various velocities, the system needs a means of leveraging this information to reason about the cost of the terrain ahead of it. We can predict the mean roughness μ_R and variance v_R of a cell in the BEV map given its feature and the velocity of the vehicle.

$$\mu_R, v_R = p(R|O, S) \tag{5.9}$$

This is accomplished by modeling the accumulated data as a Gaussian process, computing the mean and variance through the following equations:

$$k(x_1, x_2) = \exp\left(-\frac{1}{2}(x_1 - x_2)^\top \Theta^{-2}(x_1 - x_2)\right) \tag{5.10}$$

$$\mu_R = k(X^*, X)k(X, X)^{-1}Y \tag{5.11}$$

$$v_R = k(X^*, X^*) - k(X^*, X)k(X, X)^{-1}k(X, X^*) \tag{5.12}$$

where k is a radial basis function (RBF) kernel with lengthscale Θ. X is a $n \times (f + 1)$ matrix containing n training samples, each consisting of an f-dimensional feature as well as its associated velocity. Y contains the associated labels (in this case roughness) for each training sample, and X^* contains the the current observations from the BEV map, along with the current vehicle velocity, that we wish to infer a roughness for.

Note that this formulation also provides an estimate of the variance, which in turn means that the final roughness prediction can be tuned with CVaR based on the user's preferred risk tolerance similar to other works ([1, 5]), where the risk-adjusted predicted roughness assuming a Gaussian distribution becomes:

$$R = \mu_R + v_R \frac{\phi(\Phi^{-1}(\alpha_R))}{1 - \alpha_R} \tag{5.13}$$

Where ϕ is the standard normal probability density function (PDF), Φ is the normal cumulative density function (CDF), and α_R is set by the user to vary risk-tolerance.

The experience buffer can also be similarly leveraged to predict speedmaps that dictate the upper-bound speed that the robot should travel for each cell. The user can specify a maximum desired roughness R_{max}, and the mean speed μ_S and variance v_S can then be predicted.

$$\mu_S, v_S = p(S|\ O, R = R_{max}),\ 0 \leq R_{max} \leq 1 \tag{5.14}$$

Algorithm 2 Speedmap Risk Adjustment

Input: Current velocity v, current roughness r, smoothing constant β, max speed from speedmap v_{max}, velocity margin τ, CVaR cutoff α_S, risk increment factor ϵ
$v_{history} \leftarrow v_{history} + \beta(v - v_{history})$
$r_{history} \leftarrow r_{history} + \beta(r - r_{history})$
if $|v_{history} - v_{max}| < \tau$ **then**
    if $r_{history} < R_{max}$ **then**
        $\alpha_S \leftarrow \alpha_S + \epsilon$
    else if $r_{history} > R_{max}$ **then**
        $\alpha_S \leftarrow \alpha_S - \epsilon$
    end if
end if
return α_S

Since we don't want to exceed R_{max}, the predicted value can be used as a speed limit for the downstream controller. Then, equations 5.10, 5.11, 5.12, can be computed again but with X, X^*, Y rearranged correspondingly to predict speed instead of roughness.

A potential issue arises with predicting speeds for out-of-distribution situations. For example, consider a scenario in which the vehicle has only driven on trails at low speeds and therefore has experienced a max roughness that is significantly below R_{max}. If this data is then used to predict a speed limit, we will likely not obtain the true value due to lack of experience. We account for this mismatch by using CVaR with parameter α_S similarly to the approach for the costmaps with α_R, but dynamically adapting α_S instead of having it set by the user. While the vehicle is traveling within some margin of the speed limit but experiencing an average roughness that is significantly less than the expected roughness R_{max}, α_S is incrementally increased, and decreased if the it is exceeding R_{max} (outlined in Alg. 2). This allows the robot to explore higher speeds until it obtains evidence that supports its predictions, and adjust its limits if encounters previously unseen terrain that is too rough.

5.3.4 One-Shot Costmap Augmentation

The above sections describe a framework through which a system can leverage features from visual foundation models to quickly learn costmaps without human labels and

adapt them with online experience. However, this formulation using supervision signals such as roughness comes with the limitation that the robot can only learn about what it can physically drive on. This means reasonable predictions cannot be made on BEV features that correspond to lethal objects.

We address this problem by relaxing the requirement of zero human labels to one human label (per type of lethal object). In the scope of this work, the primary lethal objects encountered are trees. We find that by simply choosing a single feature from the BEV map that corresponds to a tree and permanently keeping it in the buffer associated with a high roughness, we can obtain high cost values for all the trees that the robot experiences, without needing to train the network further or have a user label several trees.

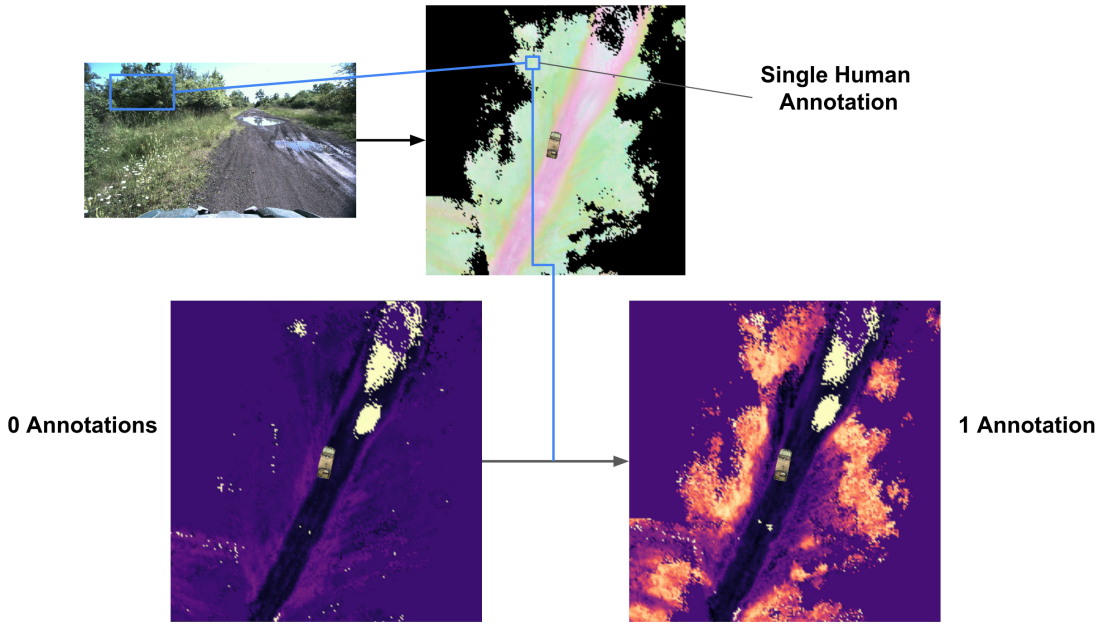

Figure 5.8: To avoid lethal terrain that can't be learned about through experience (such as trees), high cost can be manually assigned with a single annotation.

5.4 Uncertainty Avoidance

We observe that learned methods, such as in Triest et al. ([1]) as well as our own, can be effective at detecting distinct geometric obstacles and evaluating the traversability of different types of terrain, and offline results indicate generalizability to held-out

test sites. However, even with leveraging our visual BEV mapping as an input, we noticed that various out-of-distribution objects that don't resemble terrain didn't trigger high-cost predictions as expected. This is undesirable, as in practice we found some of these objects to be dangerous (e.g. a stray tire or a pallet full of nails). While this may not be the case every time, within the scope of this work we adopt a policy of avoiding these objects if the robot can find low-cost terrain around it. To achieve this behavior, we implement an additional costmap layer that acts as a heuristic for uncertainty, outlined in the following sections.

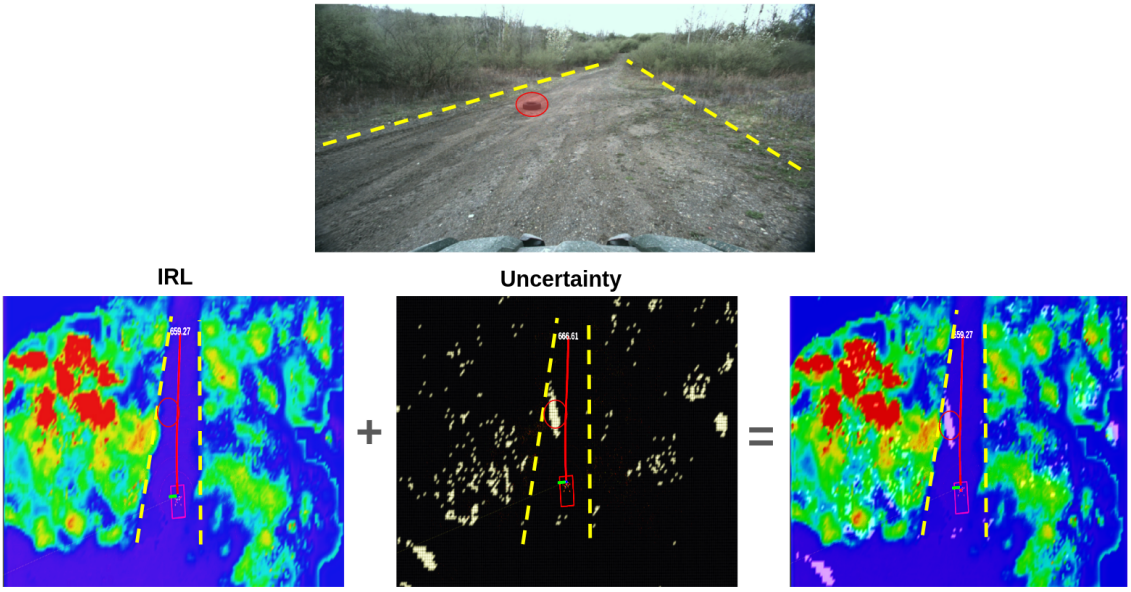

Figure 5.9: Scenario in which the vehicle must stick to the trail while avoiding a tire to the left. The IRL costmap from [1] alone correctly costs the edges of the trail and trees but does not detect the tire. By adding an uncertainty layer, the tire can be costed as lethal.

5.4.1 Quantification

In order to avoid foreign objects in an image, we require a way of assigning per-pixel uncertainty without explicitly labeled samples. We perform an approximate density estimation of a subset of training data and leverage this information to automatically detect regions of high epistemic uncertainty. We then randomly sample C-dimensional embeddings from all generated feature images from a training set, and use K-Means

clustering to generate k feature clusters.

We take inspiration from the approach in AnyLoc, in which the incoming visual features are compared to the pre-defined clusters F to generate VLAD descriptors (see 5.2.3). We leverage the residuals between the features and cluster centers and perform the following operation to determine uncertainty U.

$$u = \min_{k \in \{1,2,\ldots,K\}} \|d - F_k\|_1, \quad U = \begin{cases} 0 & \text{if } u < \tau_U \\ u & \text{otherwise} \end{cases} \tag{5.15}$$

This operation is done in the map space after the visual features have already been projected out. If the minimum distance of a pixel feature d to any cluster exceeds a threshold τ_U, we assume its corresponding terrain/object is uncertain and therefore assign it a high cost, allowing us to avoid anomalies without any further training. Additionally, this approach is less computationally intensive compared to other popular approaches such as using ensembles ([1] or training a network to predict uncertainty ([31]), allowing us to run it online with minimal overhead.

5.4.2 Detection

We use visual foundation models, specifically DINOv2, as feature extractors as we observe that the resulting clusters correspond roughly to commonly encountered terrain (e.g. gravel, dirt, bushes, grass). We use the ViT-B backbone to comply with latency requirements, and we chose the features from the 10th layer value facet (instead of query, key, or token) based on insights from AnyLoc. It is important to note that the feature image is 14x smaller than the original input image, which is why the detections in Fig. 5.10 appear to be coarse. However, we find its powerful generalizability to compensate for this shortcoming.

Table 5.2: Uncertainty Parameters

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

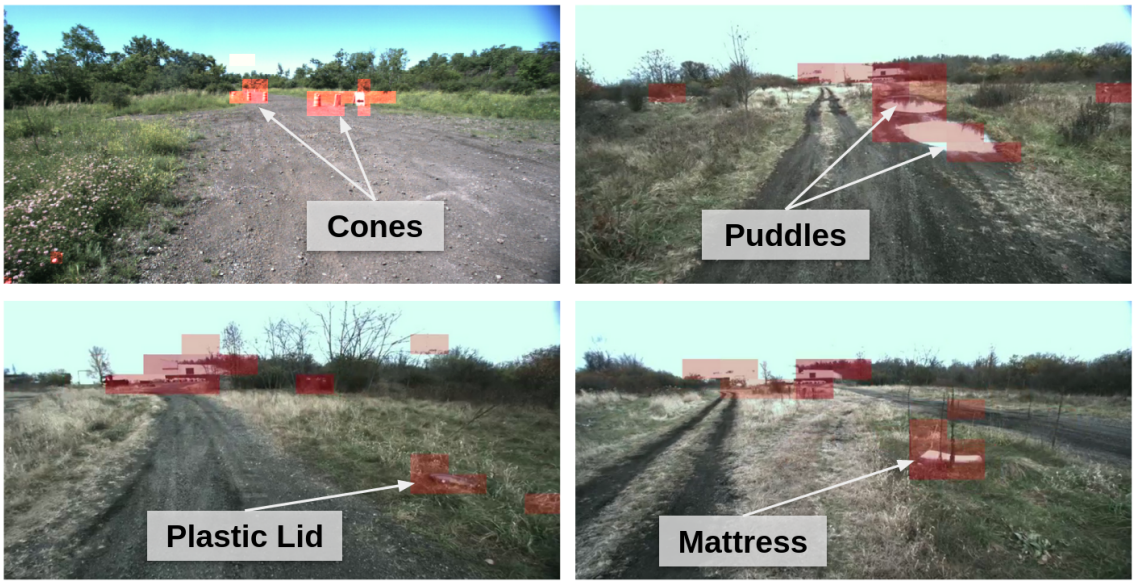

Figure 5.10: Examples of objects detected as being uncertain. Of note is the detection of objects that would be difficult with geometric features, such as items hidden in grass, items far in the distance, and puddles.

In order to compute the uncertainty value described above, we use the parameters described in Table 5.2.

Chapter 6

Experimental Results

Through our experiments we aim to show how uncertainty avoidance and adaptive costmaps and speedmaps can improve autonomous navigation. In order to evaluate its performance, we additionally run four other methods as baselines to compare against:

- **Geometric cost function**: we use the cost function use in training ALTER [24] as a geometry-informed baseline, as it is more expressive than using a simple height-threshold approach. This information is computed by aggregating a registered pointcloud using odometry, and performing geometric analysis such as terrain estimation and planarity in order to compute the required information for the cost function. We compute a speedmap that encourages higher speed in smoother areas (derived using the planarity feature).
- **Semantic segmentation using GANav [60]**: we use GANav, a state-of-the-art semantic segmentation network as a vision-informed baseline. We map the semantic logits using our visual mapping pipeline and then map the semantic classes to hand-tuned costs and speeds.
- **How Does It Feel? (HDIF) [2]**: we compare against HDIF as it is a learning-based method that heavily inspired this work. We use an updated version with internal changes designed to be more efficient and predict higher costs for geometric obstacles. It predicts a speedmap by using a pre-defined roughness threshold, querying the network with different speed inputs, and

finding the max speed-per-cell that doesn't violate the threshold.

- **Visual + Geometric Inverse Reinforcement Learning (VG-IRL)**: we compare against an extension of the inverse reinforcement learning (IRL) approach implemented by Triest et al. [1]. It has been modified to leverage visual features from DINOv2, using our visual mapping pipeline alongside the existing geometric features. It also computes a speedmap by predicting a distribution of speeds for each cell, trained from the same demonstration data that the costmap was trained on.

6.1 Improving Learned Costmaps with Out-of-Distribution Avoidance

We test our uncertainty avoidance on two robots in a number of different challenging scenarios. In this section, when present, red stars denote the starts and goals for the experiments, the black line denotes a nominal straight-line path, and the pink line represents the resulting behavior using our method.

6.1.1 Qualitative Analysis

Vehicle #1 - Yamaha ATV

We test our method on the Yamaha ATV to see how it affects navigation performance. We observe that the integration of this additional cost allows the robot to avoid objects that are different from the typical experienced terrain. As shown in Fig. 6.1, it is able to distinguish between objects and the tall-grass in proximity, succesfully supplementing the IRL costmaps.

Vehicle #2 - Clearpath Warthog

To test generalizability, we set up short-scale experiments using a ClearPath Warthog at a separate testing site containing different terrain characteristics such as dense forrests and paved roads. The VLAD clusters were generated using only data from the TartanDrive data collection site, meaning that we ran our method without any

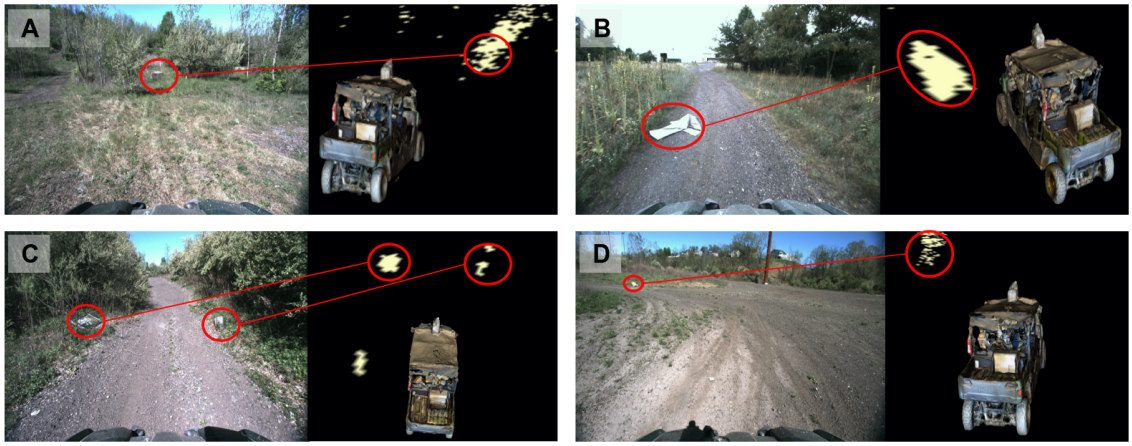

Figure 6.1: Example detections in the map space of our uncertainty avoidance deployed on the Yamaha ATV. (a): a concrete barrier hidden in tall grass. (b): a stray tarp. (c): metal debris on both sides of the trail. (d): half of a flattened traffic cone.

prior info of the new site. The primary sensors used on this platform are a Ouster OS1-64 lidar and FLIR Blackfly S camera in contrast to the Velodyne VLP-32C and Multisense S21 used on the Yamaha ATV. We use an existing stack [70] to produce lidar-based costmaps, and add ours as an additional cost, allowing the robot to avoid objects that the original stack alone would not have detected. The behavioral differences can be seen in Fig. 6.2.

6.1.2 Quantitative Results

To isolate the effect of the epistemic uncertainty detection, we designed a smaller course where the robot had to navigate to a goal directly straight ahead, with various out-of-distribution obstacles (two tires, a flattened sign, and a tarp) placed in its path (Fig. 5.4). We evaluate the number of corrections for each method over three trials, the results of which are shown in the last column of Table 6.1. We find that the additional uncertainty layer adds additional cost to these obstacles without adding cost to previously seen terrain like trails and tall grass and reduces the total number of interventions.

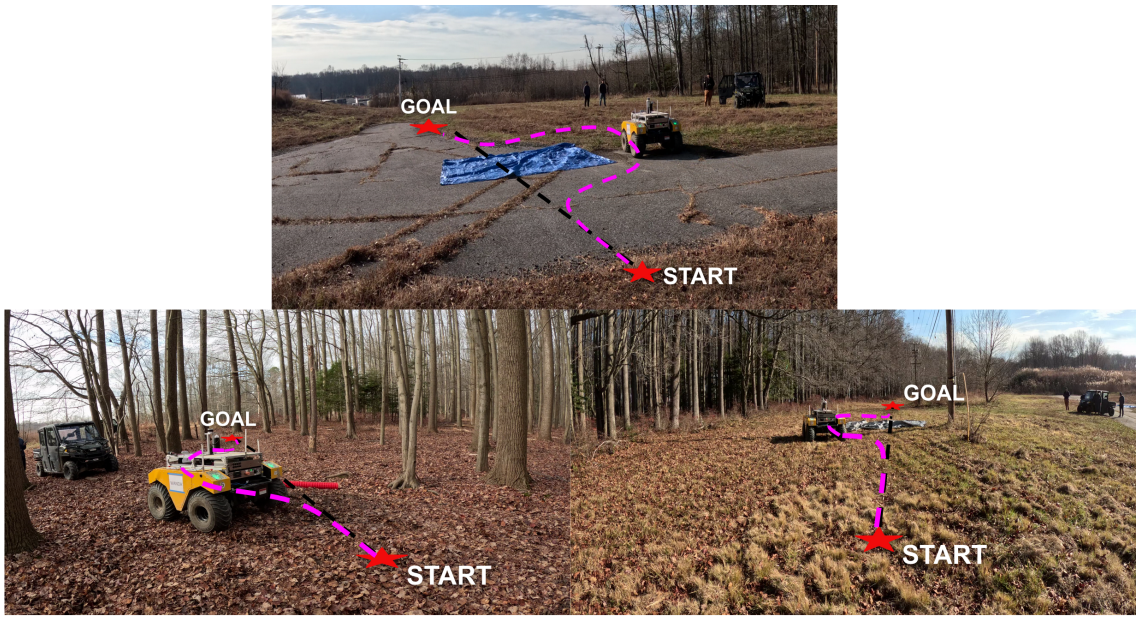

Figure 6.2: Example results of our uncertainty avoidance deployed on a Clearpath Warthog robot. By adding it as an additional layer in the existing stack, we enable the Warthog to plan around out-of-distribution obstacles in a number of different scenarios without any training data from the area.

Table 6.1: Uncertainty Avoidance Hardware Results

Method	Corrections
Geometry	5
GA-Nav	5
VG-IRL	5
VG-IRL + Unc	**1**

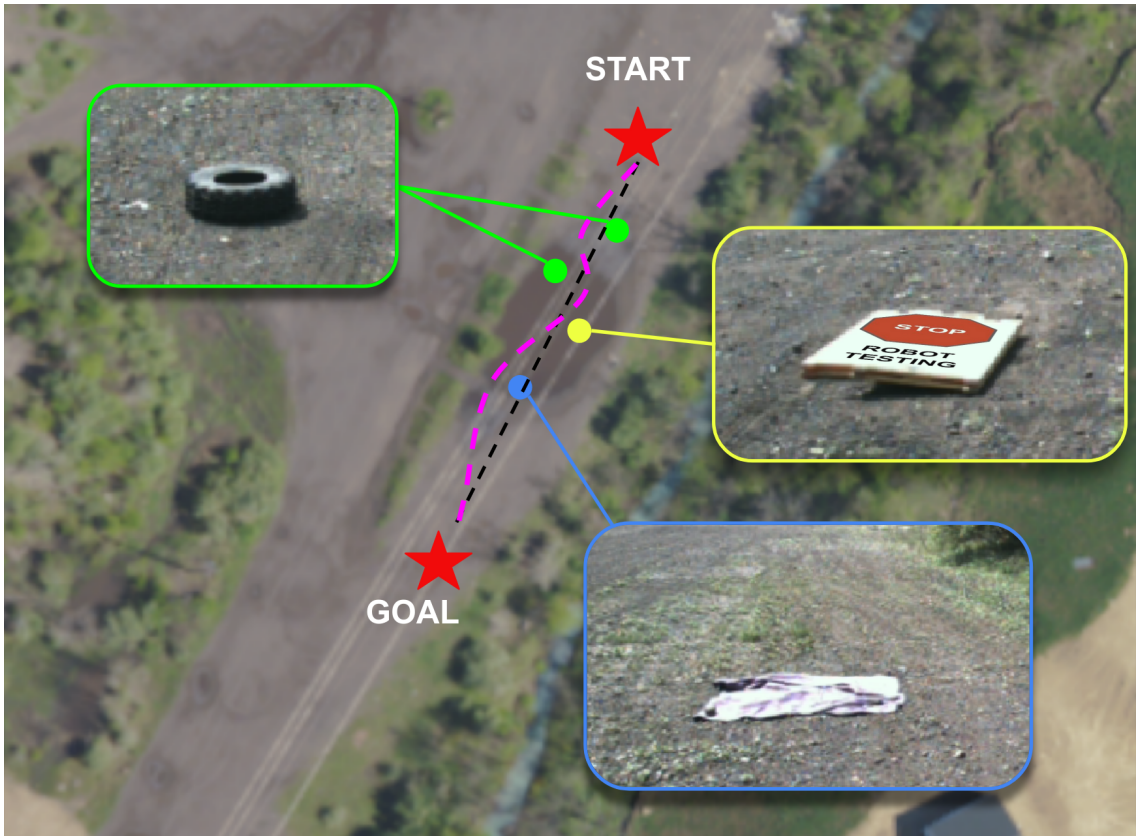

Figure 6.3: The obstacle course used to test our uncertainty avoidance against the baselines, with obstacles consisting of tires, a tarp, and a sign that says "STOP: Robot Testing". Note that all obstacles are short and geometrically insignificant enough that simple approaches such as height thresholding cannot correctly cost them without also costing traversable terrain such as tall grass.

6.2 Online Learning of Costmaps and Speedmaps

We wish to demonstrate the potential of adaptive costmaps and speedmaps in improving large-scale off-road autonomous driving. In order to do so, we run a set of experiments aiming to answer the following questions:

- Do our costmaps pick up on nuanced details (such as gravel versus smooth trail) better than other methods?
- Can our system perform general navigation tasks better than the simple baselines, and at a level comparable to state-of-the-art methods that leverage large amounts of data offline?
- Does the adaptation component lead to improved behavior and decreased roughness over time?

6.2.1 Courses

We design two navigation courses to test the performance of the costmap/speedmap/uncertainty prediction module, some using waypoint spacing of 50m, and others using manually-set waypoints with spacing of up to 150m in order to test the behavior of the system when provided with less guidance. These courses are outlined in Fig. 6.4, with course 1 being used to compare our method against other baselines and course 2 to demonstrate the large-scale navigation capabilities. We choose a separate area of the test site from which to collect training data (needed to generate the VLAD clusters).

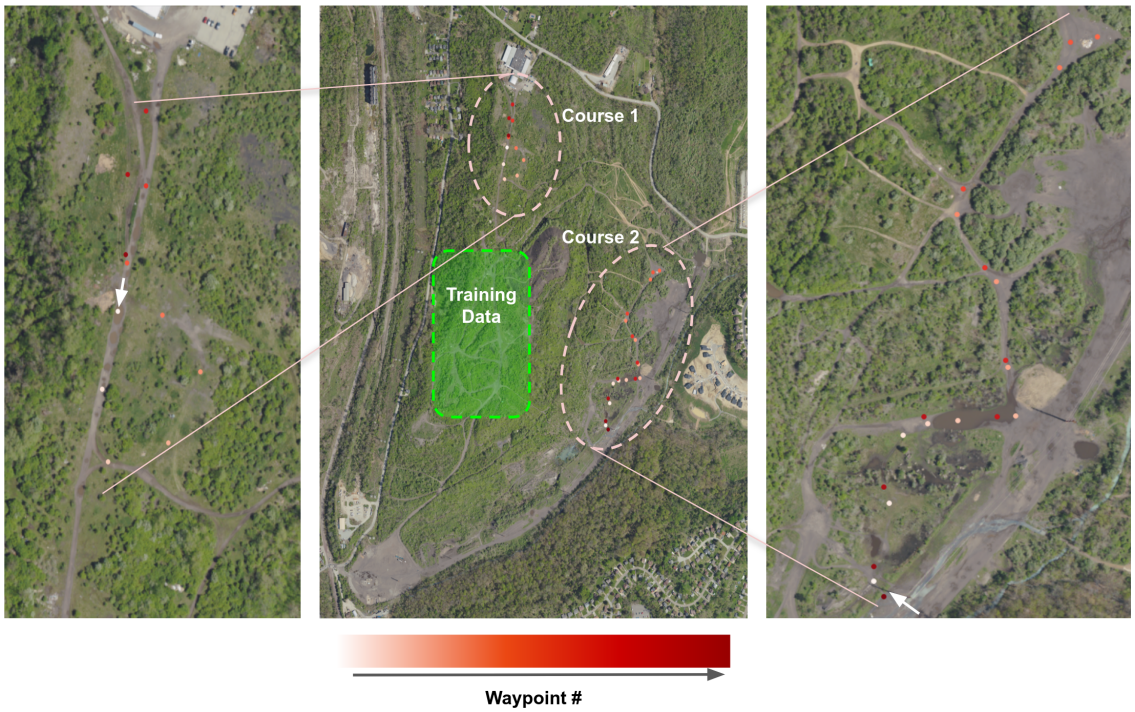

Figure 6.4: We run two navigation courses, one with waypoints spaced 50m apart and the other with manually-placed waypoints as far as 150m apart.

6.2.2 Do our costmaps pick up on nuanced details better than other methods?

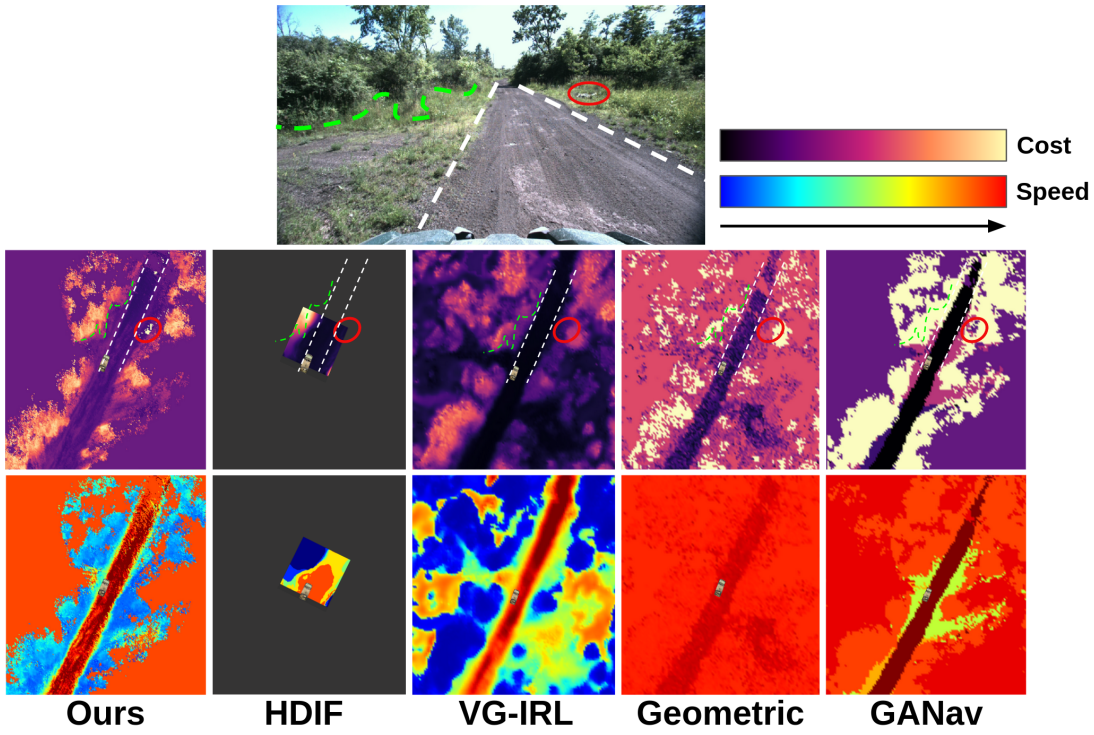

Figure 6.5: Comparison of our method agains the other baselines. Note the ability of our method to distinguish the tree line (green dashed line), trail (white dashed line), and the shattered TV hidden in the bushes (red circle). Note that all plots share the same normalization such that colors can be compared directly across all methods.

We highlight an exemplar scenario for our method in Fig. 6.5. Within less than 60 seconds of experience, it appears to outperform the simpler baselines (GANav and geometric cost function) and predict maps at a level of detail more similar to VG-IRL. Note that VG-IRL also uses geometric features from lidar as an input which provides a wider field-of-view, explaining why it contains more information.

We refer the reader back to Fig. 5.4 for a more in-depth look at how our velocity-conditioned costmaps change with input speed and how our speedmaps vary based on the user-set roughness limit. Below, we present additional qualitative examples from course 2 that demonstrate our ability to not only distinguish high-level terrain

types — such as trees, grass, and trail — but also pick up on visual cues such as grass density and gravel in order to make expressive predictions. The top left images depict the camera view, the right images depict a costmap or speedmap and the bottom left image overlays the two on top of each other to better demonstrate spatial alignment of the terrain and its corresponding cost.

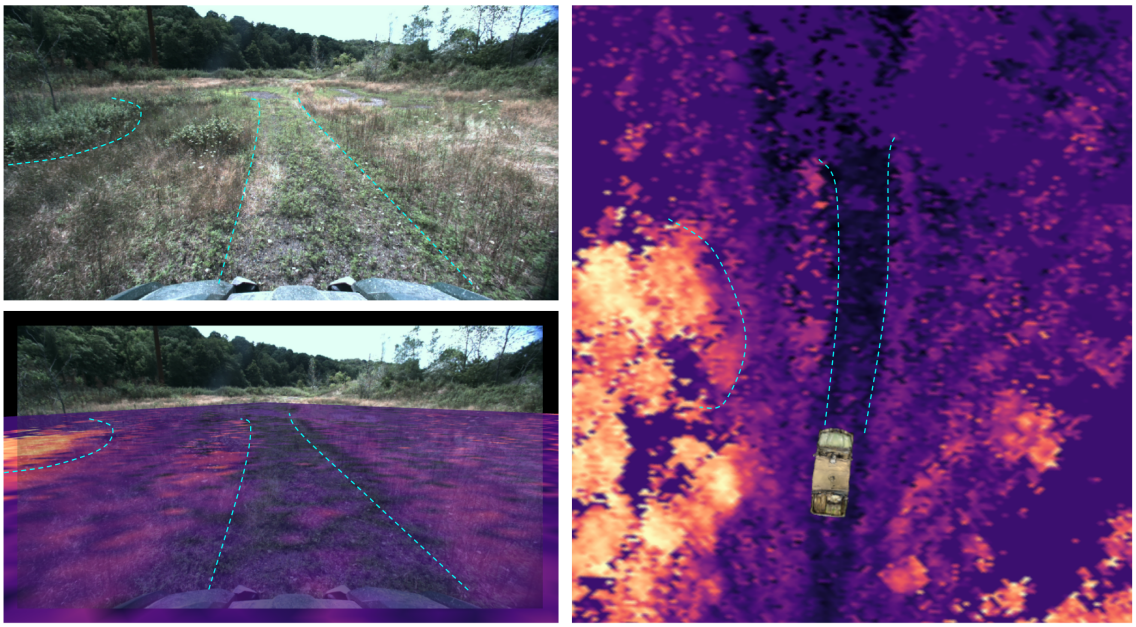

Figure 6.6: Example scenario from course 2. While much of the area is traversable, there is a clearly preferable path ahead, which is picked up by our costmaps.

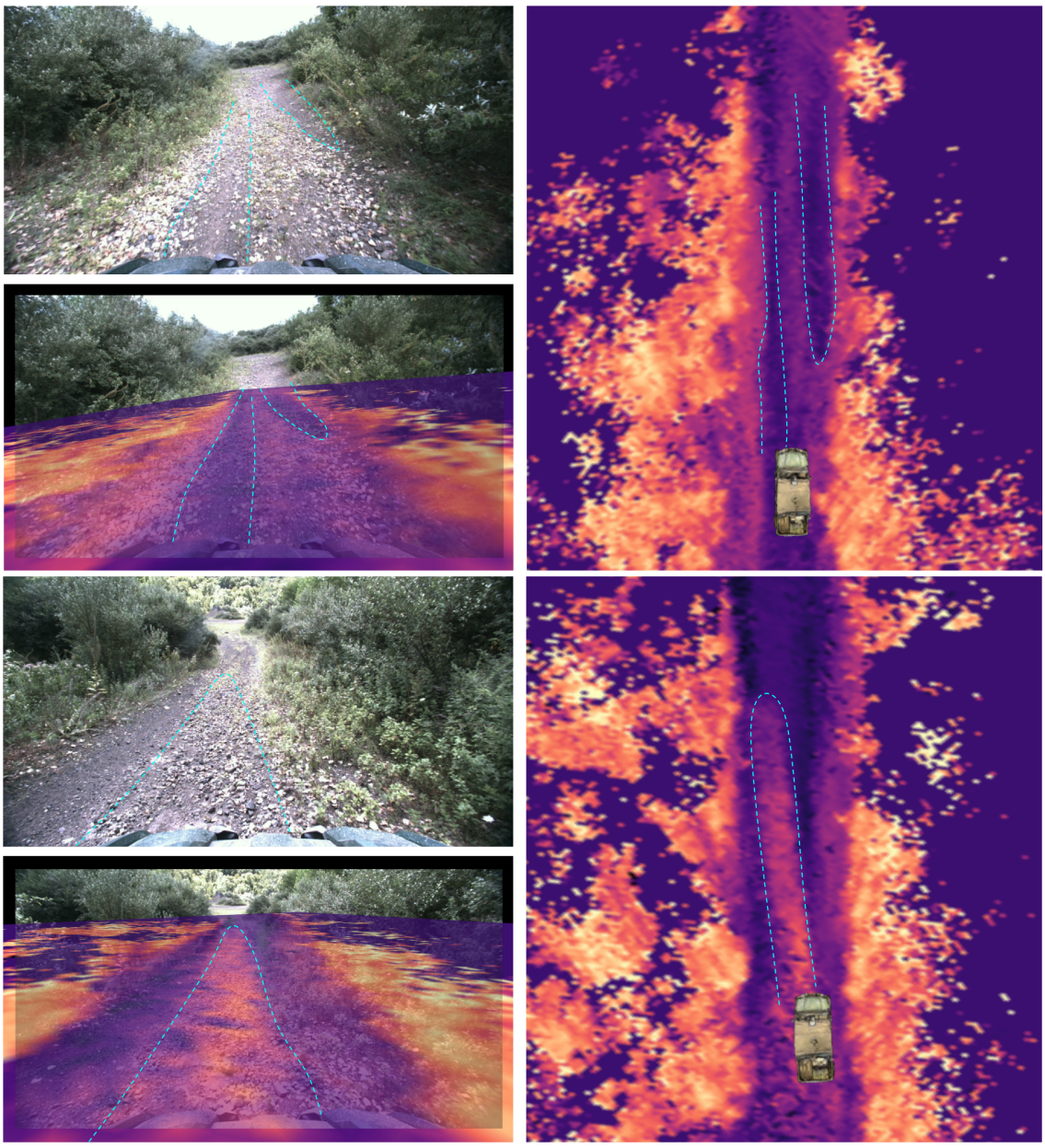

Figure 6.7: Another scenario from course 2, where the robot first drives up a rocky hill (top) then later comes back down(bottom). In both cases, our method is able to detect the difference in the rocky and smooth terrain, denoted by the dashed blue lines. Moreover, the rocky area has a higher cost during the second traversal due to the additional experience gained after the first traversal.

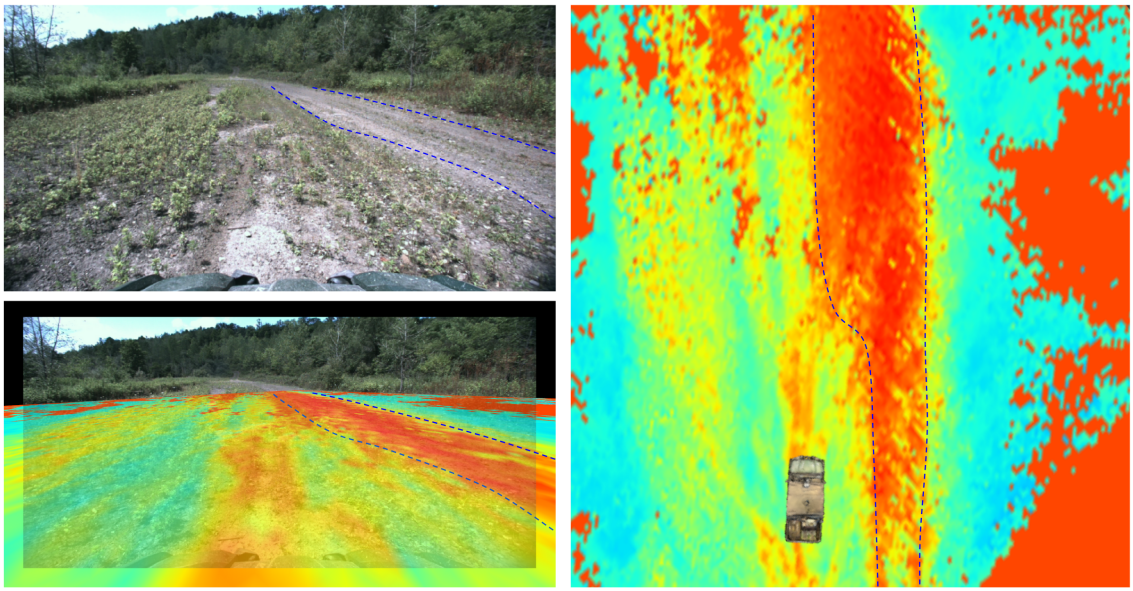

Figure 6.8: An example from course 2 demonstrating the quality of our speedmaps. Our method correctly predicts that the robot can drive faster in the smooth trail on the right (blue dashed lines) than the rough area that it currently is driving in.

6.2.3 Can our system perform general navigation tasks better than the baselines?

We run all baselines as well as our own method for three laps on course 1, evaluated on four metrics shown in Table 6.2. We count the number of times the safety driver intervened and stopped the vehicle in order to prevent damage, as well as the number of "undesirable behaviors" where the driver didn't need to intervene but the system could have taken a better route (for example driving through a rough patch of grass when there is a trail right next to it). We also record the average speed, and the proportion of the lap that the vehicle spent experiencing roughness above the user-set threshold. Note that due to field testing complications, the first three baselines (denoted by an asterisk) had to be evaluated offline in a process where we replayed sensor data and ran the autonomy stack to see what details were being captured by perception and where the vehicle was planning. While we don't guarantee the full accuracy of this evaluation, we believe it to be sufficient alongside qualitative comparison until the experiments can be redone.

Table 6.2: Navigation Performance Metrics

Method	Lap #	# Interventions	# Undesirable Behavior	Avg. Speed (m/s)	Proportion of Time Above R_{max}
Geometry*	1	3	1	–	–
	2	3	1	–	–
	3	3	2	–	–
GANav*	1	3	1	–	–
	2	5	2	–	–
	3	3	2	–	–
HDIF*	1	4	2	–	–
	2	5	3	–	–
	3	4	3	–	–
VG-IRL	1	2	4	**4.36**	**0.42**
	2	3	2	**4.60**	0.46
	3	**2**	2	**4.23**	0.40
Ours	1	**1**	**0**	4.32	0.45
	2	**0**	**1**	3.96	**0.37**
	3	**2**	**1**	3.81	**0.35**

Our method outperforms the baselines in all metrics apart from average speed. While VG-IRL had a higher average speed it also had more interventions, some of which may have been from driving too fast. Due to the fragility of our vehicle we prioritize number of interventions over speed, but we recognize that there is an inherent trade-off between the two and preferences may vary based on robot and operator.

6.2.4 Does the adaptation component lead to improved behavior and decreased roughness over time?

To better observe the adaptive behavior of our method, we run our method on course 1 for three laps. As shown in Fig. 6.9, our method can pick out trees and trail, but believes that the grass is a higher cost than it should be. Moreover, the speeds that it was predicting in the first lap resulted in an overall roughness that was exceeding the desired threshold. After finishing the first lap and coming back to the same area a second time, it has learned a better cost for the grass as well as a more appropriate

Table 6.3: Capability Comparison Against Baselines

Method	Self Supervised	Online Adaptation	Risk Aware	OOD Detection	Velocity Informed
Geometry	✓	✗	✗	✗	✗
GA-Nav	✗	✗	✗	✗	✗
HDIF	✓	✗	✗	✗	✓
VG-IRL	✓	✗	✓	✗	✓
Ours	✓	✓	✓	✓	✓

Table 6.4: Costmap Parameters for each Method

Method	Range (m)	Resolution (m)
Geometry	40	.5
GA-Nav	40	.5
HDIF	12	.1
VG-IRL	40	.5
Ours	25	.2

speed. This behavior is also demonstrated in Table 6.2, where with each consecutive lap the robot spends a lower proportion of time in high-roughness areas. We also show this across the whole course in 6.10.

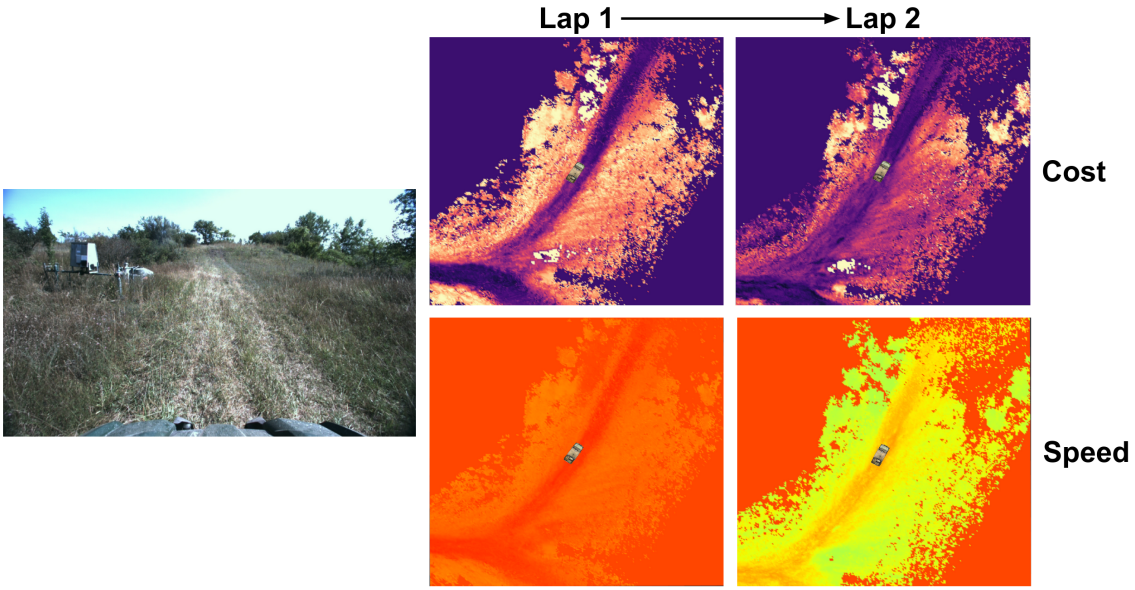

Figure 6.9: Our method is able to adapt its understanding of the environment as it collects new experiences. During the first lap of course 1, it incorrectly perceives the grassy areas as being high cost and at the same time believes it can traverse them at high speeds. By the time it reaches the same area in the second lap it has learned that the cost isn't as high as it originally predicted, and also that it needs to drive slower in those areas.

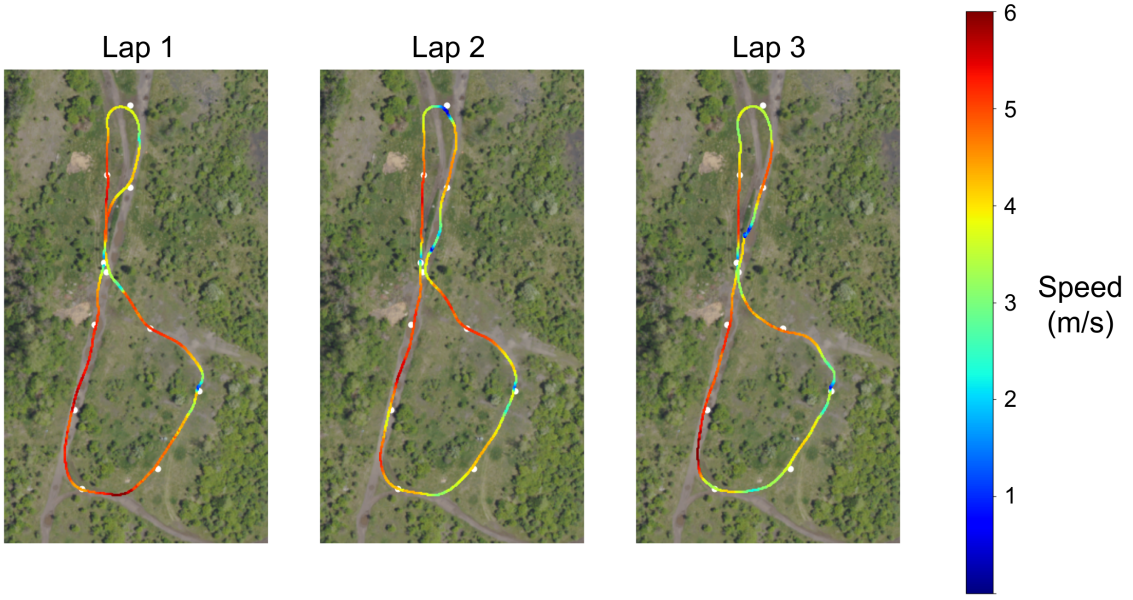

Figure 6.10: The resultant trajectories using our method, colored by speed. Over the course of three laps, the system learns to drive slower in rougher areas while maintaining high speeds on the main trails. Waypoints are colored in white.

Chapter 7

Conclusions

7.1 Summary

In this work we present a method that can predict costmaps and speedmaps that allow a system to adapt in real time to novel experiences while avoiding wildly out-of-distribution objects. This is done by leveraging generalizable features from visual foundation models, and associating them with roughness as a proprioceptive signal. This results in a system that can make more nuanced predictions about its environment than the prior state-of-the-art that in turn allow for navigation behaviors that reduce the roughness experienced by the vehicle. Further, we release a large-scale dataset designed with self-supervision in mind in order to lower the barrier-to-entry to exploring learned methods for off-road driving.

7.2 Limitations and Future Work

7.2.1 Limitations

While we improve on many of the issues presented in our baselines, there still remains a number of limitations. For example, the costmap prediction performance is heavily dependent on the type of distribution chosen as the model. For example, using Gaussian processes allows us to adapt quickly by eliminating the need for online training (at the cost of computation complexity). However, works such as [3] show

that the actual distribution isn't always uni-modal. This could be addressed by instead learning a categorical distribution, with the caveat that it would need to be trained online. Additionally, our strategy for speedmap prediction assumes that roughness increases relatively monotonoically with speed. While we observe this to be mostly the case in our system, there could exist other systems built in a way such that certain terrain is actually less rough at high speeds.

There could also be improvement in our metric evaluation process, a challenge that seems to be common across several works. Many prior works in self-supervised off-road autonomy evaluate themselves in one (or multiple) of a few limited ways. Success rate is informative in that it is agnostic to the cost-learning method, but it is a coarse signal that doesn't effectively capture more nuanced details and finer-grained behaviors. Average speed is more detailed, but doesn't always lead to clear conclusions when coupled with other metrics. For example, it's generally more desirable for the robot to drive faster, but what if it comes at the cost of more interventions? There is a strong need for a unified approach to benchmarking off-road navigation performance so that researchers can fairly evaluate their approaches.

7.2.2 Multi-Task Self-Supervised Off-Road BEVFusion

Our method is also limited by the visual mapping pipeline that it uses as a backend. Mapping at both high range and high resolution starts to become computationally infeasible to run at real-time. Moreover, our current method relies on lidar points to project image features into the BEV space, leading to a sparsity of information at range. Finally, the image features themselves are limited by the size of the model (e.g. we use the "big" version of DINOv2 instead of "giant") and the resolution of the input image. A more powerful model that can also run at high resolution in real-time should in theory dramatically improve navigation performance.

One way to address all these issues together could be by taking inspiration from [11] and training a BEVFusion [71] model that effectively learns to mimic the visual mapping pipeline. By training offline, we can generate the high-fidelity inputs and maps that we can't obtain online and use them as supervision for the network.

Bibliography

[1] Samuel Triest, Mateo Guaman Castro, Parv Maheshwari, Matthew Sivaprakasam, Wenshan Wang, and Sebastian Scherer. Learning risk-aware costmaps via inverse reinforcement learning for off-road navigation, 2023. (document), 1.2.2, 1.3, 2.2, 4.1, 4.6.4, 4.3, 5.3.3, 5.4, 5.9, 5.4.1, 6

[2] Mateo Guaman Castro, Samuel Triest, Wenshan Wang, Jason M. Gregory, Felix Sanchez, John G. Rogers, and Sebastian Scherer. How does it feel? self-supervised costmap learning for off-road vehicle traversability. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 931–938, 2023. 1.2.2, 1.3, 2.2, 4.1, 4.4.2, 4.6.1, 4.3, 5.3.1, 5.3.1, 5.3.2, 6

[3] Xiaoyi Cai, Siddharth Ancha, Lakshay Sharma, Philip R. Osteen, Bernadette Bucher, Stephen Phillips, Jiuguang Wang, Michael Everett, Nicholas Roy, and Jonathan P. How. Evora: Deep evidential traversability learning for risk-aware off-road autonomy, 2024. 1.2.2, 2.2, 7.2.1

[4] Xiangyun Meng, Nathan Hatch, Alexander Lambert, Anqi Li, Nolan Wagener, Matthew Schmittle, JoonHo Lee, Wentao Yuan, Zoey Chen, Samuel Deng, Greg Okopal, Dieter Fox, Byron Boots, and Amirreza Shaban. Terrainnet: Visual modeling of complex terrain for high-speed, off-road navigation, 2023. 1.2.2, 1.3, 2.2, 4.1, 4.6.1, 4.6.2

[5] Anushri Dixit, David Fan, Kyohei Otsu, Sharmita Dey, Ali-Akbar Agha-Mohammadi, and Joel Burdick. Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation; results from the darpa subterranean challenge. *Field Robotics*, 4(1):182–210, January 2024. 1.2.2, 1.3, 5.3.3

[6] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22:46 – 57, 07 1989. 1.3

[7] Daniel Maturana, Po-Wei Chou, Masashi Uenoyama, and Sebastian Scherer. Real-time semantic mapping for autonomous off-road navigation. In Marco Hutter and Roland Siegwart, editors, *Field and Service Robotics*, pages 335–350, Cham, 2018. Springer International Publishing. 1.3, 4.1

[8] 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. 1.3

[9] Peter Fankhauser, Marko Bjelonic, C. Dario Bellicoso, Takahiro Miki, and Marco Hutter. Robust rough-terrain locomotion with a quadrupedal robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5761–5768, 2018. 1.3, 2.2

[10] Philipp Krüsi, Paul Furgale, Michael Bosse, and Roland Siegwart. Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments: Driving on point clouds. *Journal of Field Robotics*, 34, 12 2016. 1.3

[11] Jonas Frey, Shehryar Khattak, Manthan Patel, Deegan Atha, Julian Nubert, Curtis Padgett, Marco Hutter, and Patrick Spieler. Roadrunner - learning traversability estimation for autonomous off-road driving, 2024. 1.3, 2.2, 7.2.2

[12] Matthew Sivaprakasam, Parv Maheshwari, Mateo Guaman Castro, Samuel Triest, Micah Nye, Steve Willits, Andrew Saba, Wenshan Wang, and Sebastian Scherer. Tartandrive 2.0: More modalities and better infrastructure to further self-supervised learning research in off-road driving tasks, 2024. 1.4

[13] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 2.1, 4.1

[14] Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado, Lubing Zhou, Holger Caesar, Oscar Beijbom, and Abhinav Valada. Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking. *IEEE Robotics and Automation Letters*, 7(2):3795–3802, 2022. 2.1, 4.1

[15] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2.1, 4.1

[16] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aur'elien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive

motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9710–9719, October 2021. 2.1, 4.1

[17] Amar Kulkarni, John Chrosniak, Emory Ducote, Florian Sauerbeck, Andrew Saba, Utkarsh Chirimar, John Link, Marcello Cellina, and Madhur Behl. Racecar – the dataset for high-speed autonomous racing, 2023. 2.1

[18] Jean-François Tremblay, Travis Manderson, Aurélio Noca, Gregory Dudek, and David Meger. Multimodal dynamics modeling for off-road autonomous vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1796–1802, 2021. 2.1

[19] Matthew Sivaprakasam, Samuel Triest, Wenshan Wang, Peng Yin, and Sebastian Scherer. Improving off-road planning techniques with learned costs from physical interactions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 2.1, 2.2

[20] Maggie Wigness, Sungmin Eum, John G Rogers, David Han, and Heesung Kwon. A rugd dataset for autonomous navigation and visual perception in unstructured outdoor environments. In *International Conference on Intelligent Robots and Systems (IROS)*, 2019. 2.1, 4.1, **??**, **??**

[21] Suvash Sharma, Lalitha Dabbiru, Tyler Hannis, George Mason, Daniel W. Carruth, Matthew Doude, Chris Goodin, Christopher Hudson, Sam Ozier, John E. Ball, and Bo Tang. Cat: Cavs traversability dataset for off-road autonomous driving. *IEEE Access*, 10:24759–24768, 2022. 2.1, 4.1

[22] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. ViNT: A Foundation Model for Visual Navigation. *arXiv pre-print*, 2023. 2.1, 4.3

[23] Samuel Triest, David D. Fan, Sebastian Scherer, and Ali-Akbar Agha-Mohammadi. Unrealnet: Learning uncertainty-aware navigation features from high-fidelity scans of real environments, 2024. 2.2

[24] Eric Chen, Cherie Ho, Mukhtar Maulimov, Chen Wang, and Sebastian Scherer. Learning-on-the-drive: Self-supervised adaptation of visual offroad traversability models. 2023. 2.2, 4.1, 4.6.1, 6

[25] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017. 2.2

[26] Anelia Angelova, Larry H. Matthies, Daniel M. Helmick, and Pietro Perona. Learning and prediction of slip from visual information. *Journal of Field Robotics*, 24, 2007. 2.2

[27] Edmond DuPont, Carl Moore, Emmanuel Collins, and Eric Coyle. Frequency response method for terrain classification in autonomous ground vehicles. *Autonomous Robots*, 24:337–347, 05 2008. 2.2, 5.3.1

[28] Matías Mattamala, Jonas Frey, Piotr Libera, Nived Chebrolu, Georg Martius, Cesar Cadena, Marco Hutter, and Maurice Fallon. Wild visual navigation: Fast traversability learning via pre-trained models and online self-supervision, 2024. 2.2

[29] Anuj Pokhrel, Aniket Datar, Mohammad Nazeri, and Xuesu Xiao. Cahsor: Competence-aware high-speed off-road ground navigation in se(3), 2024. 2.2

[30] Xinjie Yao, Ji Zhang, and Jean Oh. Rca: Ride comfort-aware visual navigation via self-supervised learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7847–7852, 2022. 2.2

[31] Junwon Seo, Taekyung Kim, Seongyong Ahn, and Kiho Kwak. Metaverse: Meta-learning traversability cost map for off-road navigation, 2024. 2.2, 5.3.1, 5.4.1

[32] Sanghun Jung, JoonHo Lee, Xiangyun Meng, Byron Boots, and Alexander Lambert. V-strong: Visual self-supervised traversability learning for off-road navigation, 2024. 2.2

[33] David D. Fan, Ali-akbar Agha-mohammadi, and Evangelos A. Theodorou. Learning risk-aware costmaps for traversability in challenging environments. *IEEE Robotics and Automation Letters*, 7(1):279–286, January 2022. 2.2

[34] Dongshin Kim, Jie Sun, Sang Min Oh, J.M. Rehg, and A.F. Bobick. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 518–525, 2006. 2.2

[35] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary Bradski. Self-supervised monocular road detection in desert terrain. 08 2006. 2.2

[36] John Mai. System design, modelling, and control for an off-road autonomous ground vehicle. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, July 2020. 3.1

[37] Shibo Zhao, Hengrui Zhang, Peng Wang, Lucas Nogueira, and Sebastian 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. 3.3.1, 4.4.3

[38] Samuel Triest. Learning models and cost functions from unlabeled data for off-road navigation. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, April 2023. 3.3.3

[39] Samuel Triest, Matthew Sivaprakasam, Sean J. Wang, Wenshan Wang, Aaron M. Johnson, and Sebastian Scherer. Tartandrive: A large-scale dataset for learning off-road dynamics models. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2546–2552, 2022. 4.1, 4.3, ??, ??

[40] Peng Jiang, Philip Osteen, Maggie Wigness, and Srikanth Saripalli. Rellis-3d dataset: Data, benchmarks and analysis, 2020. 4.1, ??, ??, 5.2.2

[41] Aniket Datar, Chenhui Pan, Mohammad Nazeri, and Xuesu Xiao. Toward wheeled mobility on vertically challenging terrain: Platforms, datasets, and algorithms, 2023. 4.1, ??, ??

[42] Joshua Knights, Kavisha Vidanapathirana, Milad Ramezani, Sridha Sridharan, Clinton Fookes, and Peyman Moghadam. Wild-places: A large-scale dataset for lidar place recognition in unstructured natural environments, 2023. 4.1, ??, ??

[43] Andrzej Reinke, Matteo Palieri, Benjamin Morrell, Yun Chang, Kamak Ebadi, Luca Carlone, and Ali-Akbar Agha-Mohammadi. Locus 2.0: Robust and computationally efficient lidar odometry for real-time 3d mapping. *IEEE Robotics and Automation Letters*, pages 1–8, 2022. 4.1

[44] 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. 4.1

[45] Zhaoyuan Yang, Yewteck Tan, Shiraj Sen, Johan Reimann, John Karigiannis, Mohammed Yousefhussien, and Nurali Virani. Uncertainty-aware perception models for off-road autonomous unmanned ground vehicles. *CoRR*, abs/2209.11115, 2022. 4.1

[46] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020. 4.1

[47] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision,

2023. 4.1, 5.2.2

[48] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020. 4.1

[49] Matthew Sivaprakasam, Samuel Triest, Mateo Guaman Castro, Micah Nye, Mukhtar Maulimov, Cherie Ho, Parv Maheshwari, Wenshan Wang, and Sebastian Scherer. Tartandrive 1.5: Improving large multimodal robotics dataset collection and distribution. In *ICRA2023 Workshop on Pretraining for Robotics (PT4R)*, 2023. 4.3

[50] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. Tartanvo: A generalizable learning-based vo. *Conference on Robot Learning (CoRL)*, 2020. 4.4.1

[51] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 4.5.1

[52] Jean-François Tremblay, Martin Béland, François Pomerleau, Richard Gagnon, and Philippe Giguère. Automatic 3d mapping for tree diameter measurements in inventory operations. *arXiv preprint arXiv:1904.05281*, 2019. **??**, **??**

[53] Robin Schmid, Deegan Atha, Frederik Schöller, Sharmita Dey, Seyed Fakoorian, Kyohei Otsu, Barry Ridge, Marko Bjelonic, Lorenz Wellhausen, Marco Hutter, and Ali akbar Agha-mohammadi. Self-supervised traversability prediction by learning to reconstruct safe terrain, 2022. 4.6.2

[54] Norbert Markó, Tamás Szirányi, and Áron Ballagi. Terrain depth estimation for improved inertial data prediction in autonomous navigation systems. In *2023 IEEE International Automated Vehicle Validation Conference (IAVVC)*, pages 1–6, 2023. 4.3

[55] Shubhra Aich, Wenshan Wang, Parv Maheshwari, Matthew Sivaprakasam, Samuel Triest, Cherie Ho, Jason M. Gregory, John G. Rogers III au2, and Sebastian Scherer. Deep bayesian future fusion for self-supervised, high-resolution, off-road mapping, 2024. 4.3

[56] Parv Maheshwari, Wenshan Wang, Samuel Triest, Matthew Sivaprakasam, Shubhra Aich, John G. Rogers III au2, Jason M. Gregory, and Sebastian Scherer. Piaug – physics informed augmentation for learning vehicle dynamics for off-road navigation, 2023. 4.3

[57] Zhipeng Zhao, Bowen Li, Yi Du, Taimeng Fu, and Chen Wang. Physord: A neuro-symbolic approach for physics-infused motion prediction in off-road driving, 2024. 4.3

[58] Jonathan Yang, Catherine Glossop, Arjun Bhorkar, Dhruv Shah, Quan Vuong, Chelsea Finn, Dorsa Sadigh, and Sergey Levine. Pushing the limits of cross-embodiment learning for manipulation and navigation, 2024. 4.3

[59] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. GNM: A General Navigation Model to Drive Any Robot. In *International Conference on Robotics and Automation (ICRA)*, 2023. 4.3

[60] Tianrui Guan, Divya Kothandaraman, Rohan Chandra, Adarsh Jagan Sathyamoorthy, Kasun Weerakoon, and Dinesh Manocha. Ga-nav: Efficient terrain segmentation for robot navigation in unstructured outdoor environments. *IEEE Robotics and Automation Letters*, 7(3):8138–8145, 2022. 5.2.2, 6

[61] Nikhil Keetha, Avneesh Mishra, Jay Karhade, Krishna Murthy Jatavallabhula, Sebastian Scherer, Madhava Krishna, and Sourav Garg. Anyloc: Towards universal visual place recognition. *IEEE Robotics and Automation Letters*, 2023. 5.2.2, 5.2.3

[62] Mike Ranzinger, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Am-radio: Agglomerative vision foundation model reduce all domains into one. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12490–12500, June 2024. 5.2.2

[63] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. 5.2.2

[64] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 5.2.2

[65] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, 2010. 5.2.3

[66] David Stavens and Sebastian Thrun. A self-supervised terrain roughness estimator for off-road autonomous driving, 2012. 5.3.1

[67] Lorenz Wellhausen, Alexey Dosovitskiy, René Ranftl, Krzysztof Walas, Cesar Cadena, and Marco Hutter. Where should i walk? predicting terrain properties from images via self-supervised learning. *IEEE Robotics and Automation Letters*,

4(2):1509–1516, 2019. 5.3.1

[68] Mateo Guaman Castro. Self-supervised costmap learning for off-road vehicle traversability. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, August 2023. 5.3.1

[69] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967. 5.3.1

[70] ARL autonomy stack, 2022. Accessed: 2024-07-13. 6.1.1

[71] Zhijian Liu, Haotian Tang, Alexander Amini, Xingyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. 7.2.2