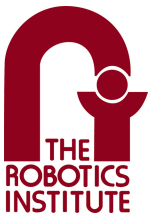
Self-Supervised Costmap Learning for Off-Road Vehicle Traversability

Mateo Guaman Castro

CMU-RI-TR-23-44

July 21, 2023



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

Thesis Committee:
Professor Sebastian Scherer, *chair*
Professor Deepak Pathak
Professor Yonatan Bisk
Gokul Swamy

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

To my parents and brother, for their unwavering love and support.

Abstract

Estimating terrain traversability in off-road environments requires reasoning about complex interaction dynamics between the robot and these terrains. However, it is challenging to build an accurate physics model, or create informative labels to learn a model in a supervised manner, for these interactions. We propose a method that learns to predict traversability costmaps by combining exteroceptive environmental information with proprioceptive terrain interaction feedback in a self-supervised manner. Additionally, we propose a novel way of incorporating robot velocity in the costmap prediction pipeline. We validate our method in multiple short and large-scale navigation tasks on a large, autonomous all-terrain vehicle (ATV) on challenging off-road terrains, and demonstrate ease of integration on a separate large ground robot. Our short-scale navigation results show that using our learned costmaps leads to overall smoother navigation, and provides the robot with a more fine-grained understanding of the interactions between the robot and different terrain types, such as grass and gravel. Our large-scale navigation trials show that we can reduce the number of interventions by up to 57% compared to an occupancy-based navigation baseline in challenging off-road courses ranging from 400 m to 3150 m.

Acknowledgments

First of all, I would like to thank my advisor, Basti. When I first came to CMU, I knew two things: I wanted to learn more about machine learning for robotics, and I wanted to work on exciting, real robots. I found both at the AirLab. Over my two years working with him, he gave me the opportunity to fulfill both of these goals and work on fun projects. I am thankful to Basti for giving me the freedom to pursue my interests, the support and infrastructure to work on a real robot platform, and the encouragement to give back to my communities. One thing that I really admire about Basti is that he always encourages students to do service in the lab and in the department, and that he cares about the development of his students as both people and researchers. He is a true example of kindness in academia. Thank you for the support over the past two years!

I am grateful to my thesis committee for all the academic and non-academic discussions we have had. I am very grateful to have been taught most of the things I know about deep learning for computer vision and robot learning by Deepak, and I feel grateful and lucky to have had the opportunity to discuss research and non-research related things over the last years with someone who I considered one of my research heroes before starting at CMU. I learned from him a lot about being fearless at pursuing big ideas, and executing them well to produce impactful research. Much of the work in this thesis is motivated by his research vision. I also want to thank Yonatan for not only research advice, but also grad school advice. His energy is contagious, and he has an incredible gift of always providing insightful comments, directions to pursue, and advice. Finally, I would like to thank Gokul for giving me an example of the type of graduate student and researcher I want to be. He is a walking encyclopedia, a kind person, an incredibly hard worker, an excellent communicator (I hope to match his slide making abilities at some point!), and a great friend. I have learned a lot from our discussions of the type of research that is worth pursuing.

Over my last two years at CMU, I learned that robotics is a team effort, and I would like to thank the team that allowed this work to happen in the first place. First, I would like to thank Wenshan Wang for effectively being a second advisor, for taking me under her wing and providing incredible high and low level support during my time at CMU, and for helping me

celebrate my victories. I would also like to thank Sam Triest for being a code wizard, helping with different parts of the system, for the fun times field testing in Gascola and in Graces Quarters, and for being a good friend. I would not have been able to do all of this work without him. I also want to thank the other members of the team, Matthew Sivaprakasam, Andrew Saba, Cherie Ho, Charles Noren, Parv Maheshwari, Shubhra Aich, and Micah Nye, for insightful discussions and shared infrastructure. I would like to also thank Steve Willits, and William Pridgen from AirLab, and Jake Lammott from NREC for their help and support during our many field experiments in Gascola, as well as Jason Gregory, John G. Rogers III, Felix Sanchez, and Ian Lancaster for their support during field tests at ARL.

Working in a lab where everyone works on a real robot, where I routinely see convoys of wheeled and legged robots roaming in the office, where I can find over 25 drones a couple of steps away, and where I have access to a full-sized off-road vehicle in which I sit in while I let it autonomously drive me has left a very clear message in my head: you need to show it on a robot to show it works. The breadth and depth of expertise in this lab never ceases to amaze me, and I am grateful to have the AirLab culture be the basis for my future endeavors. The AirLab would not be what it is without its people, and I would like to thank every one of its members. I would like to thank the staff that readily helped me with ordering stuff, handling reimbursements, and facilitating my progress. In particular, I would like to thank Nora Kazour, Amanda Axelson, Brian Hutchison, Hadley Pratt, Carly Siedt, and Dayna Larson. I would like to thank all the amazing current and past PhD students in the lab. In particular, I would like to thank Brady Moon for always being my sounding board for making good presentations, editing videos, teaching me about PR, and planning shenanigans; Rebecca Martin for helping me plan events; and Jay Patrikar, Bart Duisterhof, and Yorai Shaoul for always being fun to be around. I also want to thank the many other master's, undergraduate students, and staff from whom I learned a lot and had fun with on a daily basis. I would like to thank Andrew Saba for being a robot wizard and fighting the good fight with me to bring community back to AirLab and FRC. I would also like to thank my SQHILL office mates for all the little treats, lunches, and fun discussions we had since we moved.

I am grateful to the people that took a chance on me and shaped me into a researcher. I would like to thank Evana Gizzi, for inviting me to join her research project back in undergrad, teaching me how to interact

with people and advocate for myself, and for becoming an incredible friend. I would like to thank Jivko Sinapov, for teaching me about RL, supporting me through my early research experiences, and all the letters of recommendation. I would like to thank Guillaume Sartoretti, for teaching me to think critically and always being incredibly approachable and kind whenever I needed advice. Finally, I would like to thank Howie Choset, for teaching me how to communicate effectively and supporting me to come work with him at CMU for the first time, making me fall in love with robotics.

My experience at the Robotics Institute was shaped by being a part of the Field Robotics Center, and I can only hope to one day become as persistent, hard-working, and fearless as its director, Red Whittaker. I am privileged to have been able to hear his stories about the early days of robotics, on what it took to win the DARPA Urban Challenge, his expeditions, and field deployments in extreme conditions, all told with fierce confidence and humbleness. I am thankful to the FRC scientists, mainly George Kantor, David Wettergreen, and Basti, who supported me in bringing back FRC Tea Time, and were always encouraging in this pursuit. I also want to thank the amazing staff members of FRC, for their immense knowledge and willingness to help. In particular, I would like to thank Chuck Whittaker, Tim Angert, Jim Picard, and David Kohanbash. I deeply enjoyed hearing all of your stories at Tea Time, and also being confused and nodding along when you all mentioned countless robot names, parts, and specs that I knew nothing about. I would also like to thank the RI admins that supported me at CMU, especially Lynn Miller and BJ Fecich.

The Robotics Institute is full of smart and kind people, and it would take a lot of time to mention everyone who had a positive impact during my time here. So if we ever chatted, had a taco together during Taco Tuesdays, ate pastries together during FRC Tea Time, or hung out, know that I treasure all those times and am grateful for you. There are a couple of people who deserve a special shout out. Swapnil Pande has been an incredible, supportive friend since we first set foot in NSH A403, helping me celebrate the good things and pulling me out of the lows with long chats by the Schenley Overlook, with many fun adventures, road trips, and music jams in between. I want to thank Ben Freed for always bringing a cheerful attitude to life and always being down for any adventure. I have come to know the true meaning of exhaustion after pushing myself to my limit on a bike with Ben, but I have also had incredibly fulfiling and

fun adventures that I wouldn't have had without him. I want to thank Daphne Chen for all our incredible Moments in Time, for all the many adventures and road trips, and for always being supportive, thoughtful, and kind. I pulled a lot of long nights over the last couple of years, but they always sucked a lot less when Daphne was there, from RoboMath to ML. I would like to thank some of my closest friends at CMU who made this time fun and made me feel at home in Pittsburgh: Raunaq Bhirangi, Hans Kumar, Rohan Deshpande, Vivian Shen, Abby Breitfeld, Michelle Zhao, Ravi Pandya, Benj Jensen, Alex Stephens, Ben Eisner, Tanmay Shankar, Gokul Swamy, Mrinal Verghese, Ananya Rao, Dan McGann, Harry Freeman, Winnie Kuang, and Aarrushi Shandilya. I could list out people and memories for the next ten pages, but for the sake of brevity, I would like to thank the rest of my MSR cohort, all the people from overlapping PhD and MSR cohorts I had the fortune of meeting and interacting with, and, in general, the RI community. What an inspiring place to be a part of!

Last but most importantly, I am here today only because of the infinite, unconditional support and team effort from my family, and am deeply thankful to have them in my life. I would like to thank my mom, Amalia, for always caring for me, even from afar, and dedicating her life to making my brother and I the people we are today. I'll always try my best to make sure your efforts were not in vain. I would like to thank my dad, Luis, for instilling a deep sense of curiosity in me, of hard work, and the confidence to pursue my biggest, wildest dreams. Growing up, he used to tell me "wait until you learn about Fourier transforms, that's the coolest thing ever." Well, you are the reason that I used Fourier-related methods twice in this thesis! I would also like to thank my brother, Kevin, for always taking care of me, and for picking me up during the pandemic when my world came crumbling down. You are a true example of joy, courage, and perseverance. I would like to thank Pauli, Luna, and Dana for their joy and support, my aunts, uncles, and cousins for making sure I feel like I always have a home to come back to, and my grandparents for their example of hard work and sacrifice. No seria nada sin ustedes. I hope to make you proud!

Funding

This work was supported by by the Army Research Laboratory and was accomplished under Cooperative Agreement number W911NF-21-2-0152.

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 Off-Road Autonomy

An estimated 65% of roads in the world are unpaved [39]. This accounts for about 14 million kilometers of unpaved roads, serving primarily rural areas and developing nations. In the state of Pennsylvania alone, there exist more than 25,000 miles of unpaved roads [45]. Off-road driving is not only encountered in specific applications such as agriculture, mining, exploration, humanitarian operations, or search and rescue, but it is also the de facto mode of transportation for a large number of people to navigate from point A to point B. Additionally, off-road navigation not only refers to driving on clearly marked, yet unpaved, roads. It extends to the challenge of determining the traversability of any natural terrain, given the capabilities of a vehicle or robot.

On paved roads, autonomous driving brings the promise of increased safety, reducing emissions, improving accessibility for the mobility impaired, freeing up human driving time, improving logistics, and more [70]. If we want to provide these and additional benefits to the remaining 65% of roads, which are more equitably distributed across different populations and industries, we also need to focus our efforts towards off-road autonomy.

Off-road driving can refer to a wide spectrum of terrains and environments, ranging from well-marked, packed-dirt roads, to sticky mud or sloshy surfaces caused by melting snow. On top of the inherent challenges associated with driving over

terrains that are hard to model, these terrains are also more likely to deterioate based on environmental conditions: rain can create puddles and turn dirt into mud, snow cover and fallen leaves can significantly affect how slippery the terrain is, and vegetation can occlude the ground, potentially hiding obstacles or terrain features. The lack of artificial structure means that we cannot simply apply the same on-road autonomous driving algorithms across for off-road driving. As such, our long term goal is to develop algorithms that allow traversability in off-road environments to be learned in scalable ways, without human supervision, to enable off-road autonomous driving to keep improving continually as more data becomes available.

1.2 Contributions

The work in this thesis describes improvements to the perception pipeline for off-road autonomous navigation. The key philosophy behind this work is to allow robots to learn properties of off-road terrains directly from data, without human labels. As more data becomes available for the off-road driving task, we believe that methods that can take advantage of multiple modalities and temporal information will scale and perform better than methods that rely on human-annotated data, such as image or point cloud segmentations. To this end, we propose a method to learn traversability costmaps for off-road driving directly from the robot's experience, leveraging cross-modality supervision from inertial measurement unit (IMU) measurements. We conduct real-world experiments to validate our approach on two robot platforms, a Yamaha Viking All-Terrain Vehicle (ATV), and a Clearpath Warthog UGV. In this thesis, we present the following main contributions:

1. We present a novel method for learning to predict **continuous** traversability costmaps for off-road autonomous navigation in a self-supervised manner.
2. We demonstrate the importance of conditioning the costmaps on the robot's velocity.
3. We conduct extensive, real-world evaluation of this method on two robot platforms in a variety of outdoor environments.

The remainder of this thesis is based on the following published works:

- *How Does It Feel? Self-Supervised Costmap Learning for Off-Road Vehicle*

Traversability, Guaman Castro et al., published as a conference paper at ICRA 2023 [8] (Main Contribution).

- *Learning Risk-Aware Costmaps via Inverse Reinforcement Learning for Off-Road Navigation*, Triest et al., published as a conference paper at ICRA 2023 [60] (Background).
- *TartanDrive 1.5: Improving Large Multimodal Robotics Dataset Collection and Distribution*, Sivaprakasam et al., published as a workshop paper at the Pre-Training for Robotics workshop at ICRA 2023 [50] (Conclusion).

Chapter 2

Background

In this chapter, we provide a brief overview of the relevant parts of our autonomy stack. In particular, we describe how we formulate the autonomous navigation task in off-road driving as a trajectory optimization problem, and we introduce the subcomponents which support this formulation, such as state estimation, costmap generation, and our local controller.

2.1 Autonomy Pipeline

2.1.1 Problem Formulation

We cast off-road autonomous navigation as a trajectory optimization problem. In our formulation, a trajectory τ consists of consecutive state and action pairs (x_t, a_t) at a given time step t. We define the components of x_t and a_t below. The state at time $t+1$ after taking an action a_t from state s_t is given by the dynamics function $f(x_t, a_t)$. We then specify a goal position g in the robot's local frame. The objective is then to choose actions which optimize for the following objective:

$$\min_{a_{1:T-1}} J(x_{1:T}, a_{1:T-1})$$
$$\text{s.t. } x_{t+1} = f(x_t, a_t)$$

In order to direct where and how to reach the goal position, we define a cost function J which, if minimized, informs the robot on how to get to the goal in a particular manner. To specify a goal, we can define a cost function directly proportional to the Euclidean distance to the goal from the current state, as follows:

$$c_g = K_g||p(s_t) - g||_2$$

.

In this cost function to the goal (c_g), K_g is a scalar, $p(s_t)$ is a function that extracts the position of the robot from the state s_t, and g is the goal position.

In order to specify how to reach this goal, we use costmaps, which are a spatial representation of the ground plane in which the robot operates, represented as a grid of locations at a given resolution. Therefore, every position in the ground plane of the robot belongs to a cell in the costmap. For a given state, we can query the cost for that state's cell in a costmap $J_{map}(s_t)$.

We can then define the optimization objective as a linear combination of the goal and costmap costs, to specify where to go and how to get there:

$$J(\tau) = J(x_{1:T}, a_{1:T-1}) = \sum_{t=1}^{T}[J_{map}(s_t)] + K_g||(p(s_T) - g)||_2$$

2.1.2 State Estimation

Despite having GPS sensors in our pipeline, state estimation algorithms allow us to estimate odometry more accurately. For state estimation, we use Super Odometry [71], and since the main purpose of this thesis is not related to the specific state estimation algorithm, we refer the reader to this work for more details. State estimation is most pertinent to our costmap prediction problem in two ways:

1. It allows us to aggregate perceptual inputs from different time steps at a higher resolution by enabling us to have precise motion transforms.
2. It allows us to track goal locations more accurately using our controller.

2.1.3 Costmap Generation

A costmap is a spatial representation of 3D space that informs the costs of driving over different areas of a ground plane, at a certain resolution (e.g. we can split the ground plane into cells of area $1m^2$ and set costs for each of these cells). Traditionally, costmaps specify which cells in a 2D plane are occupied, also called occupancy maps [11, 15]. To obtain an estimate for whether a cell is occupied or not based on sensor measurements, usually a height threshold is set, where everything above the threshold is set to be occupied, and everything below is free space where the robot is free to move. Then, occupied cells are assigned high costs in a costmap, and free cells are assigned zero cost in a costmap. It is common to inflate these costmaps to encourage robots to maintain a safe distance to obstacles.

One way to make costmaps more expressive than simply denoting occupation is to assign different costs to different types of terrain. Recent work has looked into using semantic classifiers from visual data to label different regions of 3D space, and assign hand-picked costs to each of the different classes [38]. However, this relies on the assumption that there are a limited number of terrain classes and that costs are fixed for each of these classes, limiting their expressivity. In this thesis, we explore the idea of learning costmaps for terrain traversability directly from the robot's experience as sensed through proprioception sensors, such as IMU.

2.1.4 Vehicle Model

Estimating a dynamics model for off-road driving is a challenging task, given the variability of the environment and the complex robot-terrain interactions given by the nature of the robot. However, we can approximate the dynamics function of an Ackermann-steered robot using a simple kinematic bicycle model (KBM). This model does not consider at all any of the forces involved in the robot-terrain interactions, such as the effect of mud or the dampening of the robot's shocks. Yet, this model is popularly used in off-road driving as it is sufficiently effective in practice [40, 60].

The basic formulation for a KBM is the following:

$$x = \begin{bmatrix} p_x \\ p_y \\ \theta \end{bmatrix}, u = \begin{bmatrix} v \\ \delta \end{bmatrix}, \dot{x} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v\tan(\delta)}{L} \end{bmatrix}$$

In this equation, p_x and p_y correspond to position in the x and y axes, θ is the robot's yaw, v is the robot velocity, δ is the steering angle, and L is the vehicle's wheelbase.

This formulation assumes that the velocities and steering angles can be changed instantaneously. This assumption does not hold in many real-world settings due to actuator delays, so we modify the KBM to use desired velocity and steering using proprotional controllers with appropriate gains, as follows:

$$x = \begin{bmatrix} p_x \\ p_y \\ \theta \\ v \\ \delta \end{bmatrix}, u = \begin{bmatrix} v_{des} \\ \delta_{des} \end{bmatrix}, \dot{x} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \frac{v\tan(\delta)}{L} \\ K_v(v_{des} - v) \\ K_\delta(\delta_{des} - \delta) \end{bmatrix}$$

2.1.5 Controller

In this thesis, we mainly focus on model predictive path integral (MPPI) control, a sampling-based formulation of model predictive control (MPC) [66]. This provides us a controller that can handle non-convex cost functions, which fit our costmap formulation. Additionally, we can control the latency of the controller on real hardware by adjusting the number of samples used in the controller and the rollout length considered for each sampled trajectory. We refer the reader to [66] for more details on the controller.

2.2 Learning from Unlabeled Data

In our setting, we aim to learn traverability costmaps from unlabeled datasets, which are more scalable and easier to collect. In this section, we briefly describe some of the philosophies behind this goal.

2.2.1 Learning from Demonstrations

In order to collect data for the off-road setting, a human must command the vehicle in ways that already present the desiderata we would want in an autonomous driving system. In other words, humans are already great at driving off-road, and if robots exhibited the same behaviors, that would already solve a great part of the problem. The key insight is that if humans are great at driving off-road, and humans have to teleoperate the vehicle to collect data for learning-based algorithms, imitating the human behavior from this data is a reasonable approach to solving the off-road autonomous driving problem. There is extensive theory in learning from demonstrations [43, 46, 52, 55, 56], including imitation learning, learning from interventions, and inverse reinforcement learning.

In work led by my labmate Sam Triest [60], we consider the task of learning to predict uncertainty-aware traversability costmaps directly from the human teleoperations using inverse reinforcement learning. Intuitively, under this formulation, the robot learns where to drive based on the inherent preferences of the humans who collected the dataset. In order to allow the user to control the risk-tolerance levels that the autonomous robot should exhibit, we formulate uncertainty using conditional value at risk (CVaR) via an ensemble of models. Although much of the work in this literature requires learning from expert demonstrators, we empirically show that by simply learning from the potentially suboptimal preferences shown by the human teleoperators during data collection, we are able to improve off-road navigation significantly. For more details, please refer to [60] .

2.2.2 Self-Supervision and Cross-Modal Supervision

In this thesis, we are mainly concerned with learning to predict traversability costmaps by taking advantage of the multimodal and temporal nature of the data collected

by the many sensors on our robot. While exteroceptive sensors such as lidar and cameras provide rich information about the environment, proprioceptive sensors, such as IMUs and wheel encoders, provide direct measurements of the effect of the terrain on the vehicle. The main body of this thesis will describe this approach in detail.

2.3 On-Road vs Off-Road Autonomy

In this thesis, we consider the problem of autonomous off-road autonomy. While a significant amount of effort has contributed to great strides in autonomous driving in the last decades, most of these efforts have been focused on on-road driving. Many of the advances in on-road autonomous driving can be attributed to industry research and their ability to collect large amount of annotated data [6, 54, 67]. These large, annotated datasets have enabled state-of-the-art algorithms in bird's-eye-view (BEV) mapping [32, 33], 3D object detection and tracking [34, 69], and motion prediction [23, 42], among other problems. A great part of what makes on-road autonomous driving challenging is the existence of multiple agents in a shared environment — including pedestrians, cyclists, and other vehicles — as well as the extreme low-risk tolerance to ensure that autonomous vehicles do not pose risks to other road users. This makes problems such as object detection and motion forecasting important, leading to object-centric solutions to these problems, such as achieving high segmentation-accuracy for other road users so that they can be appropriately avoided.

Off-road autonomous driving poses fundamental and practical challenges that differentiate it from its on-road counterpart. Off-road autonomous driving deals with navigation in environments with a lack of artificial structure, such as roads, lanes, and expected objects in the road. Instead, in these types of environments is no longer clear where to drive, and subtle differences in the environment can drastically affect the effects on the vehicle. This means that off-road autonomous robots need to reason about the traversability properties of different types of terrain. The robot-terrain interaction is determined both by the nature of the terrain as well as the capabilities of the robot. In off-road environments, robots are bound to encounter challenging terrains such as gravel, rugged roads, vegetation, water, mud, ice, and a diverse array of smaller objects such as logs, pebbles, etc. A large, off-road side-by-side vehicle can

likely navigate over most of these terrains. An even larger tank may be able to even push down trees to make its way, while a small off-road RC car may not even be able to traverse over tennis ball-sized pebbles.

One of the main practical challenges is the lack of publicly available, large datasets to train machine learning models for off-road autonomy. In the last couple of years, some datasets for off-road driving have been released [24, 28, 51, 59, 65]. However, these datasets tend to be lacking in amount of data, diversity of locations where the data is collected, or lack of important modalities and annotations. Yet, even if we had large amounts of data, it is unclear how to annotate this data for off-road autonomy. For instance, compliant bushes and vegetation may be easily traversable, but the similar vegetation with rocks hidden underneath this vegetation may cause damage to the vehicle if traversed over. These fundamental and practical challenges exemplify the need to rethink perception, planning, and control methods for off-road autonomous vehicles. This thesis is mainly concerned with the perception aspect of the off-road autonomous driving problem.

Chapter 3

How Does It Feel? Self-Supervised Costmap Learning for Off-Road Vehicle Traversability

3.1 Introduction

Outdoor, unstructured environments are challenging for robots to navigate. Rough interactions with terrain can result in a number of undesirable effects, such as rider discomfort, error in state estimation, or even failure of robot components. Unfortunately, it can be challenging to predict these interactions a priori from exteroceptive information alone. Certain characteristics of the terrain, such as slope, irregularities in height, the deformability of the ground surface, and the compliance of the objects on the ground, affect the dynamics of the robot as it traverses over these features. While these terrain characteristics can be sensed by proprioceptive sensors like Inertial Measurement Units (IMUs) and wheel encoders, these modalities require direct contact with the terrain itself. Additionally, the robot's interaction with the ground leads to dynamic forces which are proportional to velocity and suspension characteristics. In order to *feel* what navigating over some terrain at some velocity is like, we argue that the robot must actually traverse over it.

Previous approaches for off-road traversability have focused on representing ex-

Figure 3.1: We present a costmap learning method for off-road navigation. We demonstrate the efficacy of our method on a large ATV (left), and on a Warthog UGV robot (right).

teroceptive information as occupancy maps [13, 16], or learning semantic classifiers from labeled data to map different terrains to different costs in a costmap [26, 38]. Yet, this abstracts away all the nuance of the interactions between the robot and different terrain types. Under an occupancy-based paradigm, concrete, sand, and mud would be equally traversable, whereas tall rocks, grass, and bushes would be equally *non*-traversable. In reality, specific instances of a class may have varying degrees of traversability (e.g. some bushes are traversable but not all).

Other approaches have characterized terrain roughness directly from geometric features [17, 30, 31]. Yet, what we are really interested in capturing is roughness as the vehicle experienced it, rather than capturing the appearance or geometry of roughness. For instance, a point cloud of tall grass might appear rough, but traversing over this grass could still lead to smooth navigation if the terrain under the grass is smooth. Finally, other learning-based methods learn predictive models or direct control policies for off-road navigation [25]. However, many of these do not take into account robot dynamics, which are fundamental in scenarios where the state of the robot, such as its velocity, can lead to a wide range of behaviors. For instance, speeding up before driving over a bump in the terrain can lead to jumping, whereas slowing down will usually result in smoother navigation.

In this paper, we propose a self-supervised method that predicts costmaps that

reflect nuanced terrain interaction properties relevant to ground navigation. Motivated by examples in legged locomotion [64], we approach this problem by learning a mapping from rich exteroceptive information and robot velocity to a continuous traversability cost derived from IMU data. We propose a learning architecture which combines a CNN backbone to process high-dimensional exteroceptive information with a feed-forward network that takes in a Fourier parameterization of the low-dimensional velocity information, inspired by recent advances in implicit representation learning [41, 57].

Our main contributions are: a) learning to aggregate visual, geometric, and velocity information directly into a *continuous-valued* costmap, without human-annotated labels, and b) demonstrating ease of integration into traditional navigation stacks to improve navigation performance, due to our choice of using a top-down metric representation. In the rest of this paper, we present our contributions in more detail as follows:

- We present an IMU-derived traversability cost that can be used as a self-supervised pseudo ground-truth for training.
- We demonstrate a novel way to combine low-dimensional dynamics information with high-dimensional visual features through Fourier feature mapping [57].
- We propose a system that produces continuous-valued learned costmaps through a combination of visual and geometric information, and robot velocity.
- We validate our method on outdoor navigation tasks using two different ground robots.

3.2 Related Work

There exists over a decade of work in learning methods for off-road traversability [3, 4, 10, 19, 21, 29, 53], much of which can be traced to the DARPA LAGR [22] and Grand Challenge [58] programs. These methods learned lightweight terrain traversability classifiers based on visual and geometric appearance [3, 4, 19, 21, 29, 53]. Other approaches directly estimate a terrain roughness score by analyzing planarity [30] or eigenvalues of the terrain point cloud [31].

Much of the recent literature on learning for off-road traversability estimation

has focused on semantic segmentation of visual data into discrete classes [35, 38]. However, it is not immediately obvious how to map human semantic classes into costs that can be directly used for path planning and control, which often results in hard-coded mappings. More recent work by Shaban et al. [48] aims to alleviate the need for explicit mappings from human semantic classes to costs by directly learning discrete traversability classes, such as low-cost and high-cost, in a metric space from geometric data. Yet, these approaches require a large amount of labeled semantic data for training, and lack expressiveness given the limited number of traversability classes.

Recently, other works have looked into learning predictive models, control policies, and risk-aware costmaps directly from visual and multimodal inputs for navigation in challenging off-road environments. While BADGR-based methods [25, 49] learn a *boolean* predictor for whether a specific sequence of actions will lead to a bumpy trajectory, we learn a continuous value for traversability that is aggregated into a costmap that can be used directly to optimize trajectories, without the need to query the network for every sample of our MPPI optimizer [66]. Triest et al. [59] learn a neural network-based dynamics model for a large ATV vehicle and explore the use of different types of multimodal data as input to their neural network. Sivaprakasam et al. [50] learn a dynamics model in simulation to derive a dynamics-aware cost function for downstream planning tasks. Fan et al. [12] learn a traversability risk costmap from lidar data, and Cai et al. [7] learn a speed distribution map that is converted into a costmap using a conditional value at risk (CVaR) formulation. Triest et al. [60] learn CVaR-based uncertainty-aware traversability costmaps from lidar data using inverse reinforcement learning.

Most recently, traversability estimation for off-road robots has shifted towards learning continuous costmaps in a self-supervised manner with IMU signals as learning targets, with these methods learning from RGB data [47, 64, 68], or point clouds [61], and [47, 61] conditioning on robot speed similar to our approach. We use both RGB and point cloud data in our approach to generate high-resolution, continuous costmaps, and we demonstrate our approach in large-scale, challenging off-road courses at much higher speeds on two different large robot platforms.

3.3 Costmap Learning

We now introduce our self-supervised costmap learning method, which associates a proprioception-derived cost to high-dimensional visual and geometric data, and robot velocity. In the following section, we will describe our costmap learning pipeline (Figure 3.2), which consists of:

1. Derivation of a cost function from proprioception,
2. Extraction of a map-based representation from visual and geometric data,
3. Representation of velocity as an input to our model, and
4. Training a neural network to predict traversability cost.

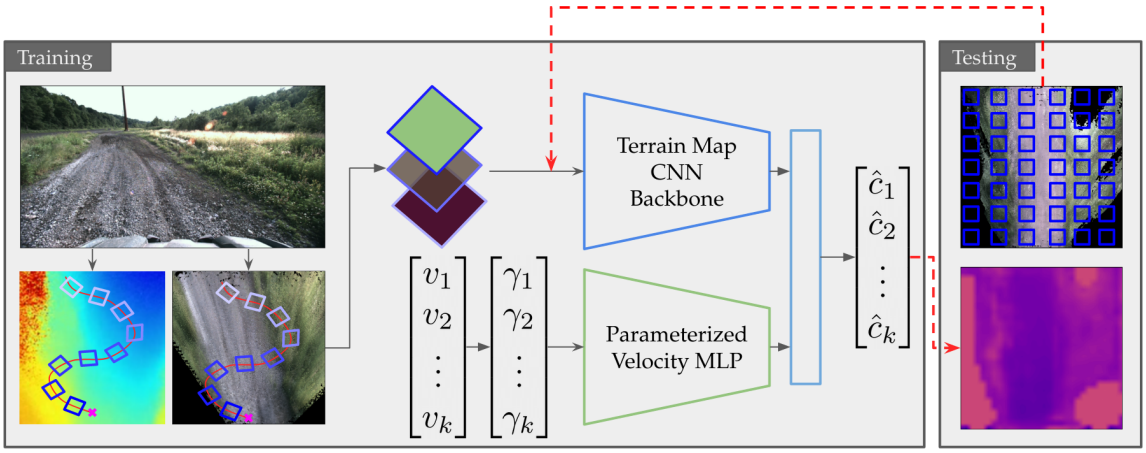


Figure 3.2: **System Overview**: During training, the network takes in patches cropped from a top-down colored map and height map along the driving trajectory, as well as the parameterized velocity corresponding to each patch. The network predicts a traversability cost for each patch, supervised by a pseudo ground-truth cost generated from IMU data. During testing, the whole map is subsampled into small patches, which are fed into the network to generate a dense, continuous costmap.

3.3.1 Pseudo-Ground Truth Traversability Cost

We aim to learn a continuous, normalized traversability cost that describes the interactions between the ground and the robot, and which can be directly used for path planning. As shown in previous work [10, 53, 64], linear acceleration in the z axis,

as well as its frequency response, capture traversability properties of the environment that not only depend on the characteristics of the ground, but also on the speed of the robot. To obtain a single scalar value that generally describes traversability properties of the terrain, such as the roughness, bumpiness, and deformability, we use the bandpower of the IMU linear acceleration of the robot in the z axis:

$$\hat{c} = \int_{f_{\min}}^{f_{\max}} S_{a_z}^{W}(f)\, df, \tag{3.1}$$

where $\hat{c}$ is our estimated traversability cost, $S_{a_z}^{W}$ is the power spectral density (PSD) of the linear acceleration a_z in the z axis, calculated using Welch's method [63], , and $f_{\min}$ and $f_{\max}$ describe the frequency band used to compute bandpower. We use a frequency range of 1-30 Hz, since this range highly correlates with human-labeled roughness scores, and normalize based on data statistics from recorded trajectories, as detailed in the Appendix.

3.3.2 Mapping

We represent the exteroceptive information about the environment in bird's eye view (BEV), which allows us to aggregate visual and geometric information in the same space, which we refer to as the "local map." We use a stereo matching network [9] to obtain a disparity image, from which we estimate the camera odometry using TartanVO [62], a learning-based visual odometry algorithm. We use this odometry and the RGB data to register and colorize a dense point cloud which we then project into a BEV local map. The local map consists of a stacked RGB map, containing the average RGB value of each cell, and a height map, containing the minimum, maximum, mean, and standard deviation of the height of the points in each cell, ignoring all points 2 meters above the ground surface to deal with overhangs. Additionally, we include a boolean mask that describes areas in the local map for which we have no information, either due to occlusions or limited field of view of the sensors.

3.3.3 Velocity Parameterization

At high speeds, rough terrain leads to higher shock sensed by the vehicle, proportional to its speed, as explored in [53]. We obtain velocity-conditioned costmaps by including

robot velocity as an input to our network.

In order to balance the high-dimensional local map input and the low-dimensional velocity input, we use Fourier feature mapping [57] on the robot's velocity. Recent advances in implicit neural representations have shown that mapping a low-dimensional vector (or scalar) to a higher dimensional representation using Fourier features can modulate the spectral bias of MLPs (which is usually biased towards low-frequency functions [44]) towards higher frequencies by adjusting the scale of the Fourier frequencies. Intuitively, we hypothesize that this parameterization lets the netwrok learn a function that more readily adjusts to subtle changes in velocity input, and prevents the network's predictions from being dominated by the high-dimensional 2D inputs. We use the following parameterization:

$$\gamma(v) = \begin{bmatrix} \cos(2\pi b_1 v) \\ \sin(2\pi b_1 v) \\ \vdots \\ \cos(2\pi b_m v) \\ \sin(2\pi b_m v) \end{bmatrix} \tag{3.2}$$

In Equation 3.2, v corresponds to the norm of the 3D velocity vector, $b_i \sim \mathcal{N}(0, \sigma^2)$ are sampled from a Gaussian distribution with tunable scale σ, and m corresponds to the number of frequencies used to map the scalar velocity value into a $2m$ dimensional vector.

3.3.4 Costmap Learning

Our costmap learning pipeline consists of three parts: a) obtaining local map patches from robot trajectories, b) training a network to predict traversability costs from a set of patches and associated pseudo ground-truth labels, and c) populating a cost map using the trained network at test time.

Local Map Patches: We extract 2x2 meter patches (roughly the robot footprint) of the local map corresponding to the parts of the environment that the robot traversed over during the dataset generation. Since the patch under the robot is not observable from a front-facing view, we first register all the local maps into an aggregated map. We use the robot odometry to locate and extract the patch in the global map at

a given 2D position and orientation. At each of these positions, we use a sliding window of the last one second of IMU linear acceleration data to obtain a pseudo ground-truth traversability cost as described in Section 3.3.1. We also record the velocity at these positions for training.

Cost Learning: We train a deep neural network $f_\theta(P, v)$ parameterized by weights θ, as shown in Figure 3.2, that takes in as input local map terrain patches $P \in P$ and corresponding Fourier-parameterized velocities $\gamma(v), \gamma : R^+ \rightarrow R^{2m}$, and predicts the traversability cost of each patch $\hat{c} = f_\theta(P, \gamma), f_\theta : P \times R^{2m} \rightarrow R^+$. We use a ResNet18 [20] backbone to extract features from the patches, and a 3-layer MLP to extract features from the parameterized velocity. Finally, we concatenate these features and pass them through a fully-connected layer with sigmoid activation to obtain a normalized scalar value representing the learned traversability cost. We train this network using a Mean Squared Error (MSE) loss between the predicted costs and the pseudo ground-truth values using the Adam [27] optimizer.

Costmap Prediction: We produce costmaps at test time by taking the current local map in front of the robot, extracting patches at uniformly sampled positions (with the same orientation as the robot's current orientation), and passing them into the network. We then reshape each of the cost predictions into a costmap that corresponds to the original local map. We find that it is important to add a stride in the sampling process to allow the patch cost querying through the network to run in real time. In our experiments, we subsample the local map with a stride of 0.2m, and upsample the reshaped predicted cost values back to the shape of the local map, which allows us to produce costmaps at 7-8 Hz on an onboard NVIDIA GeForce RTX 3080 Laptop GPU.

3.4 Experimental Results

3.4.1 Training Data

To train our network, we use TartanDrive [59], a large-scale off-road dataset containing roughly 5 hours of rough terrain traversal using a commercial ATV with a sensor suite. We use the stereo images in the dataset to obtain dense point clouds using TartanVO [62] (section 3.3.2), as well as the IMU and odometry data to obtain traversability

costs and velocities, respectively. In order to effectively train our network, we find it necessary to augment and balance the data with respect to the pseudo ground-truth traversability cost. We enforce a 2:1 ratio of high to low cost frames, resulting in 15K training frames, and 3K validation frames. Additionally, we fine-tune the base model with 9.5K training frames and 1.3K validation frames collected on the Warthog platform for our Warthog experiments.

3.4.2 Navigation Stack

We validate our learned costmaps in off-road navigation tasks, where the goal is to navigate to a target location. For state estimation, we use Super Odometry [71] on the commercial ATV and a pose graph-based SLAM system on the Warthog [18]. For path planning and control, we use model predictive path integral control (MPPI) [66]. We plan through a kinematic bicycle model with actuator limits on both the ATV and the Warthog. In order to obtain costs that can be used for the MPPI optimization objective, we query the learned costmap via Equation 3.3. This cost function queries the costmap for each state-action pair in the trajectory τ and sums it with a weighted Euclidean distance between the final state and the goal g (where $p(s)$ extracts the x-y position of state s). Since our learned costmap only learns costs for parts of the terrain it has driven over, it will not know what cost to assign to obstacles that the robot is incapable of traversing over. We alleviate this by composing our learned costmap with a lethal height costmap for obstacle avoidance (with a high threshold), resulting in costmap J_{map}. K_g is found empirically.

$$J(\tau) = \sum_{t=0}^{T} [J_{map}(s_t, a_t)] + K_g ||(p(s_T) - g)||_2 \tag{3.3}$$

3.4.3 Robot Platforms

We demonstrate our system on two different ground robots autonomously operating at 3 m/s: a large all-terrain autonomous commercial vehicle (ATV), and a Clearpath Warthog robot [1], a large unmanned ground vehicle (UGV). For more hardware details, please see our Appendix.

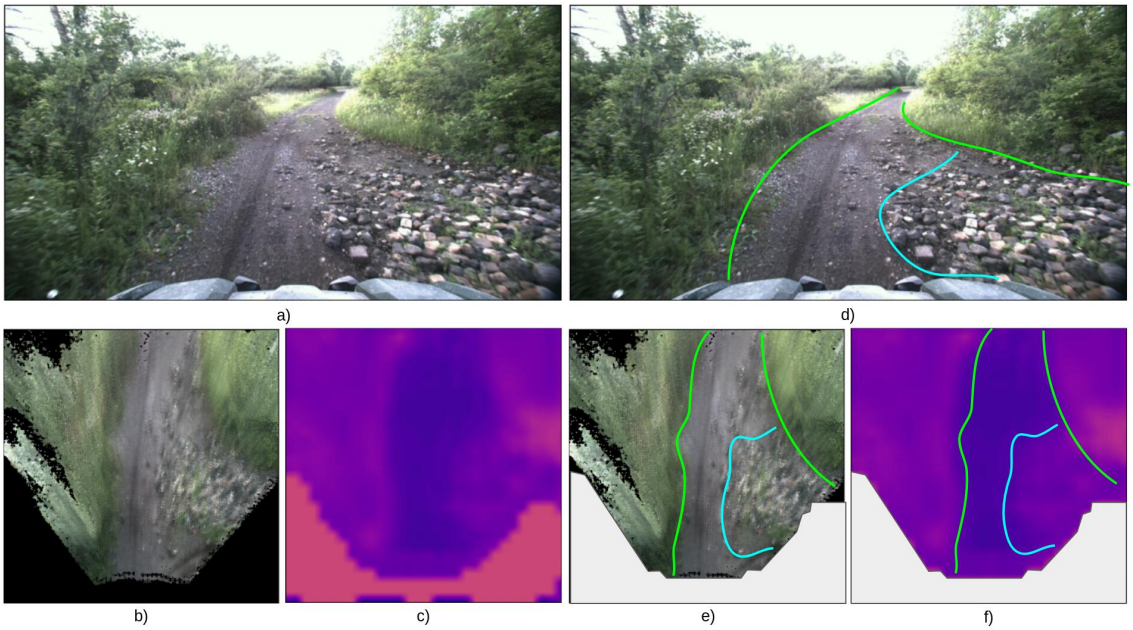


Figure 3.3: Learned Traversability Costmaps: a), b), and c) show the robot's front-facing view, the corresponding RGB map, and the predicted costmap respectively. d), e), and f) show a manually annotated version of a), b), and c) for easier visualization of the different traversability properties that our costmap captures. We set the unknown regions to have a cost of 0.5 as shown in c), which we mask out in f). Brighter shades of pink correspond to higher cost regions, and darker shades of purple correspond to lower cost regions.

3.4.4 Results

In our experiments, we are interested in answering the following questions:

1. Do our learned costmaps capture more nuance than the baseline lethal-height costmaps?
2. How much of an effect does velocity have in our predicted costmaps?
3. Do we obtain more intuitive behaviors and better navigation performance using learned costmaps inside a full navigation stack?
4. How well does our method transfer to other robots?

Roughness Estimator	Val. Loss ($\times 10^{-2}$)
$\sigma_z[14] + v$	6.08
$\sigma_z[14] + \text{RGB} + v$	5.92
Ours	**4.82**

Table 3.1: Comparison to other appearance-based roughness estimators. Our method outperforms both a geometric, as well as a geometric and visual roughness estimators.

Do learned costmaps show more nuance?

To observe trends in the learned costmap, we tele-operated the ATV around our test site and visually analyzed the predicted costmaps in different environments. We observe that our learned costmaps are able to estimate different costs for terrains of the same height but with different traversability properties. In Figure 3.3, we show a scenario with gravel, smooth dirt, and vegetation, where our costmap predicts different costs for these different terrains.

We notice some particular trends in our experiments. Transitions in texture from one terrain to another are usually discernible in our costmaps. Terrains covered in grass exhibit a higher cost than smooth dirt paths. Finally, terrains with higher frequency textures, such as large patches of gravel, are predicted to have higher costs than smooth terrains.

Does velocity affect the predicted costmaps?

We evaluate whether adding velocity as an input to our network improves the loss achieved in the validation set. We analyze three different models:

1. **patch-model**: just the ResNet backbone for patch feature extraction.
2. **patch-vel-model**: combines the ResNet backbone with an MLP to processes normalized velocity.
3. **patch-Fourier-vel-model**: combines the ResNet backbone with an MLP that processes Fourier-parameterized normalized velocity.

Training results for all three models and ablations are shown for five random seeds in the Appendix. We find that adding velocity as an input to the network leads to better performance than using local map patches alone, and that the `patch-Fourier-vel-model` performs best as measured by the validation loss. Our

ablations show that using both RGB and height statistics performs slightly better than using just RGB, and that the scale and number of frequencies used for Fourier parameter mapping of velocity do not make much of a difference.

We compare our method with two appearance-based roughness baselines to evaluate whether our network captures more accurately the robot-terrain interactions. The first baseline characterizes roughness as the standard deviation of height in a given patch [14]. We extend this metric with the average RGB values in a patch for our second baseline. For a fair comparison, we add normalized velocity as an input to both baselines. We learn the weights for a linear combination of these inputs through logistic regression. We compare with our best learned model in the validation set (Table 3.1).

We also evaluate the effect of robot speed in the predicted costmaps in a real-robot experiment. In this experiment, we command the ATV different velocities and aggregate the average predicted cost over a straight 200 m trail with similar terrain characteristics throughout. Additionally, we integrate the sum of costs of the entire costmap (energy) along the trajectory. We summarize the results in Table 3.2, and show the resulting costmaps predicted at different velocities in Figure 3.4. We find that traversability cost and overall costmap energy generally increase as robot speed increases.

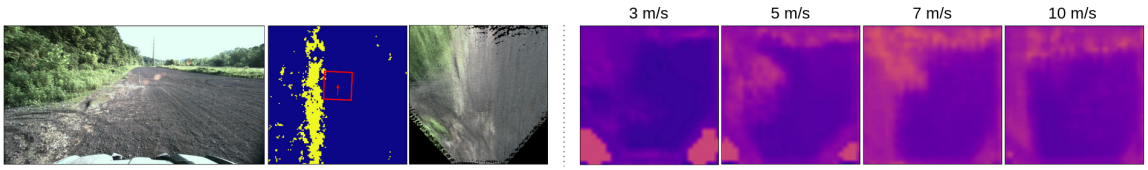


Figure 3.4: Costmaps at different velocities, as predicted by the `patch-Fourier-vel` model. The left block shows front-facing image, lethal-height costmap, and a top-down RGB map. The right block shows four costmaps of the same scenario at increasing speeds. Brighter means higher cost. Higher cost at the top of the costmaps is produced by artifacts in registration.

Do learned costmaps improve *short-scale* navigation?

We deploy our learned costmaps within a full navigation stack in two different navigation tasks, and compare the performance against a navigation stack which

Velocity	Pred. Traj. Cost	GT Traj. Cost	Costmap Energy
3 m/s	0.093	0.038	0.185
5 m/s	0.163	0.055	0.250
7 m/s	0.201	0.084	0.289
10 m/s	0.195	0.106	0.266

Table 3.2: Effect of velocity on the predicted costmap in a real-robot experiment. Higher speeds generally result in higher costs, both on the robot's trajectory and the costmap as a whole. Note that while the IMU-based ground truth cost increases monotonically as velocity increases, the predicted cost stops increasing after 7 m/s likely due to lack of enough training data at higher speeds.

uses just a baseline geometric occupancy-based costmap. When using our learned costmaps, we compose them with the occupancy-based costmap to provide basic obstacle avoidance capabilities. We set the lethal height to be 1.5 m and all unknown regions to have a cost of 0.5 for all experiments.

We set up two navigation courses for the ATV, ATV-Warehouse (Figure 3.5) and ATV-Pole, where the task is to simply move 400m and 200m straight ahead, respectively. We design these controlled courses such that a straight path would lead the robot to go over rougher terrain or patches of vegetation, but reasoning about the terrain would lead the robot to take a detour to stay on smoother trails. For ATV-Warehouse, we run five trials with our learned costmap and five trials with the baseline costmap, and for ATV-Pole, we run three trials for each costmap.

The baseline costmap leads the robot to navigate in a straight line (with some noise), which leads straight into the patch of vegetation. Additionally, this baseline stack leads to less consistent navigation, which occurs because most of the terrain in front of the robot appears to be "traversable," since there are no obstacles above the 1.5 m lethal height, which leads to many different "optimal" paths. On the other hand, our learned costmaps cause the robot to consistently navigate around the patch, staying on marked paths, and even moving away from parts of the path with gravel towards smoother sections of the trail.

We measure the cross-track error (CTE) as a way of measuring the deviation of the chosen path from a naïve, straight path to the goal. In this case, lower is *not* necessarily better, since the experiment is explicitly designed for the robot to deviate from the straight path to find the smoother trails if it is able to reason about different

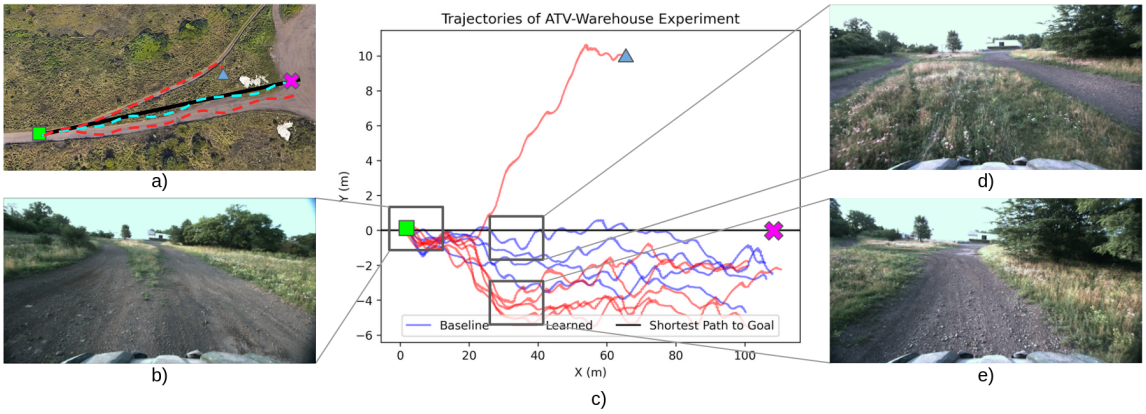


Figure 3.5: Short-scale navigation experiment (ATV-Warehouse). When using our costmap, the robot deviates from the straight path to the goal and chooses the smoother dirt path (e) over the patch of vegetation (d) to get to the goal. a) Sketch of the trajectories taken when using the baseline stack (blue) and our stack (red). The green square is the start position, and the pink cross is the goal position. One trajectory ends in an intervention (triangle) due to the robot taking an equally-smooth dirt path to avoid the grass. b), d), and e) show front-facing views at those points in the trajectory. c) shows all trajectories, as well as the direct path to the goal (black).

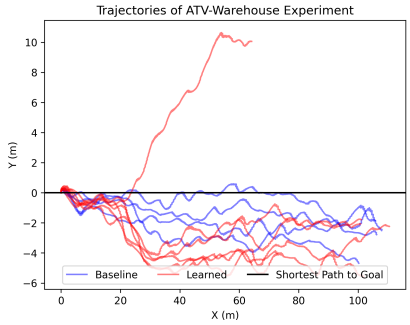
terrains. We show the preferred paths in Figure 3.5, and numerical results in Table 3.3. The results for the ATV-Warehouse experiment are shown in Figure 3.6, and the results for the ATV-Pole experiment are shown in Figure 3.7). Below, we describe these two experiments in more detail.

ATV-Warehouse: In this experiment, there was a patch of grass directly in front of the robot, and there were two smoother dirt paths at each side. As seen in Figure 3.6a, in most cases, the robot running our learned costmaps took the dirt path to the

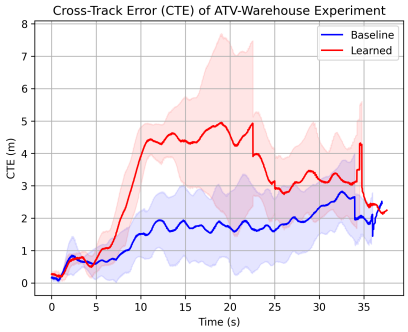
Course	Nav. Stack	Avg. CTE (m)
ATV-Warehouse	Baseline	1.29 ± 1.02
ATV-Warehouse	HDIF (Ours)	3.39 ± 0.94
ATV-Pole	Baseline	2.44 ± 2.25
ATV-Pole	HDIF (Ours)	2.68 ± 0.41

Table 3.3: Avg. Cross-track error (CTE) between planned paths and the shortest path to goal. Learned costmaps resulted in trajectories that deviated more from the shortest path to avoid areas of higher cost, e.g. grass or gravel.

right of the patch of grass. In one occasion, the robot took the path to the left. While at first this avoided the patch of grass, eventually the robot needed to cut into the patch to get to the goal. This run was stopped early due to an intervention to prevent damaging the robot. When using the baseline costmap, the robot consistently drove straight over the patch of grass. We ran the navigation course five times for each costmap. As observed from the cross track error in Figure 3.6b, the robot consistently takes a path that strays away from the nominal path to go over smoother terrain.



(a) Paths taken by the robots. Nominal trajectory (black) represents the straight path between the origin and the goal.



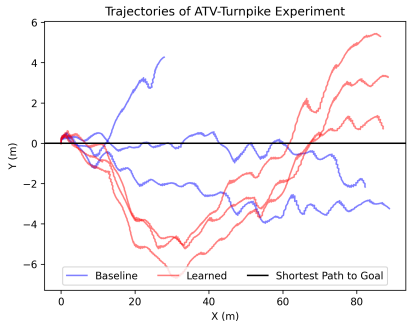
(b) Average cross-track error between the nominal trajectory and each of the paths taken by the robot over time.

Figure 3.6: ATV-Warehouse experiment.

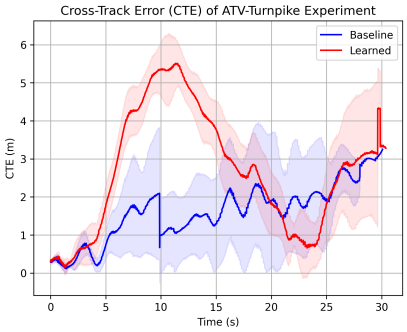
ATV-Pole: In this experiment, there was also a patch of grass in between the robot and its goal, with two dirt paths surrounding it. However, the dirt paths were also bordered with loose gravel. As seen in the supplemental video, the robot identifies both the grass and the loose gravel as having higher cost than the smooth dirt path, which leads the robot to avoid it. Figure 3.7a shows the paths that the robot took. One of the baseline runs was stopped early to prevent damage to the robot. We ran the navigation course three times for each costmap.

Do learned costmaps improve large-scale navigation?

To provide a direct measurement of navigation performance, we set up three large scale navigation experiments in three challenging off-road courses, with waypoints every 50 m, and measure our navigation performance using number of interventions, as is common practice [5, 30].



(a) Paths taken by the robots. Nominal trajectory (black) represents the straight path between the origin and the goal.



(b) Average cross-track error between the nominal trajectory and each of the paths taken by the robot over time.

Figure 3.7: ATV-Pole experiment.

Figure 3.8: Overview of large-scale navigation experiments. The leftmost figure shows a satellite view of the three courses: red, blue, and green. The top row shows sample aerial views of the courses, while the bottom row shows sample first-person robot images. The color of the rectangle around the picture denotes the course.

Red Course: The red course consists of flat forest trails of finer gravel, with vegetation on the sides of the trails. One of the main challenges of this 400 m trail is that it has a couple of tight turns, which are challenging for both the baseline and our proposed navigation stack.

Blue Course: This 3150 m course consists of flat and hilly terrain, which ranges from smooth gravel to large pebbles, notably in the sloped sections. Additionally, part of the course is covered in vegetation about 1 m tall.

Green Course: The main challenge of this 900 m trail is that about half the trail is covered in tall vegetation.

Navigation Stack	Course	Interventions	Course Length (m)
Baseline	Red	7	400
HDIF (Ours)	Red	**3**	400
Baseline	Blue	9	3150
HDIF (Ours)	Blue	**6**	3150
Baseline	Green	11	950
HDIF (Ours)	Green	**7**	950

Table 3.4: Number of interventions (lower is better) in three large-scale navigation courses with varying degrees of difficulty and different types of environment features, such as slopes, vegetation, and tight turns. Our How Does It Feel (HDIF) navigation stack achieves lower number of interventions in all courses compared to the baseline stack.

The safety driver was directed to only intervene when either: a) the robot missed a waypoint by over 4 m, or b) a collision with a non-traversable obstacle was imminent. These courses (Figure 3.8) include flat and hilly terrain, loose material ranging from fine gravel to large cobbles, and vegetation of various dimensions. For both navigation stacks, we set the target speed to 3.5 m/s, lethal height to 0.5 m and fill unknown values of our costmap to 0.5. We observe that in all three courses, our navigation stack with learned costmaps outperforms the baseline stack in terms of number of interventions, as detailed in Table 3.4. Our learned costmaps lead the robot to stay on smoother paths and avoid vegetation if possible, which leads to an overall decrease in intervention events. In practice, this leads to the robot taking wider turns, staying on clear paths, and avoiding rough features that lead to bumpy navigation, such as large cobbles. We urge the reader to watch the videos for these experiments on our website.

How well does our method transfer to other robots?

We set up two additional short-scale navigation experiments using the Warthog, with a fine-tuned model as detailed in Section 3.4.1. We used a separate geometry-based autonomy stack [2] as the baseline, and for our stack we simply composed their lidar-based binary costmap with our learned costmap. Despite the Warthog being a skid-steer robot, we treat it as if it had an Ackermann steering geometry, similar to the ATV, since its cameras only face forward.

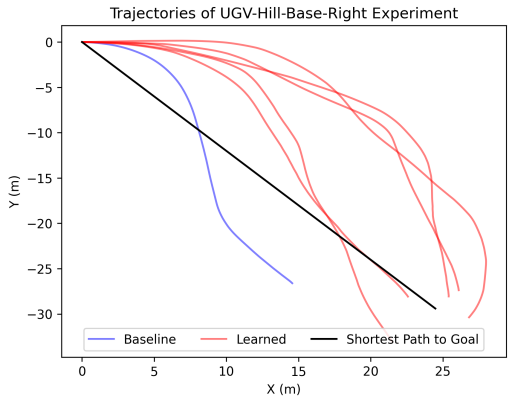
UGV-Hill-Base-Right: In this experiment (Figure 3.9a), the Warthog robot was given a GPS waypoint as a goal about 50 m diagonally to the right. This goal was located along a concrete path that surrounds a patch of vegetation. The baseline navigation stack immediately starts turning and navigates straight to the goal, going over the vegetation, since it does not reason about the difference in traversability properties between the smooth concrete and the grass. We ran a single trial using the baseline, and five using our learned costmaps. As seen in Figure 3.9b, in all five runs with our costmaps, the robot takes a longer path that deviates from the nominal straight line path, avoiding the grass and instead taking instead the smoother concrete path to the goal.

UGV-Forest-Fork: In this experiment (Figure 3.10a), the Warthog is placed before a fork in a forest trail, where both sides of the fork lead to the same spot. We artificially litter the left path of the fork with tree branches, rocks, and fallen leaves. On the other hand, we clear the path on the right of all obstacles to make it as smooth and possible. Note that none of the small obstacles on the left path would be registered as lethal obstacles using a geometry-based lethal-height binary costmap. We ran three trials with the baseline stack, and three trials with our costmap. When using our costmaps, we first move the robot left-to-right in-place manually to fill in the RGB map (since the visual range is shorter than that of the lidar-based baseline stack). We observe that in all three runs, the robot prefers the smoother path to reach the goal. The trajectories are shown in Figure 3.10b.

We observe that with the baseline stack, the robot uniformly chooses either of the paths to get to a goal at the other end of the fork in the trail. On the other hand, with our costmaps, the robot consistently chooses the clear path on the right to get to the other side of the trail. Note that a visual semantic classifier would label both of these trails as belonging to the same class, but the subtle differences in the features of the trails cause different amounts of roughness.

In both experiments, it is clear that our costmaps lead the robot to choose smoother paths by reasoning about the different traversability properties of different terrains. We urge the reader to visit our website for experiment videos.

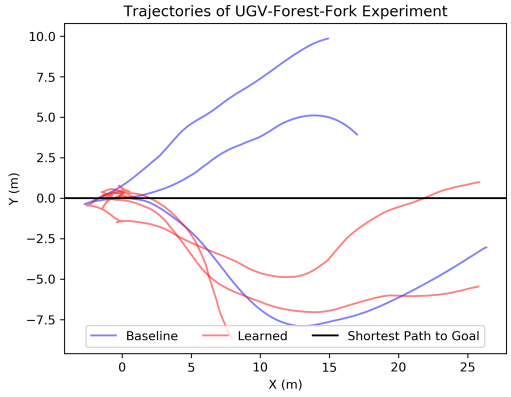
(a) UGV-Hill-Base-Right experiment, where the goal is about 50 m diagonally to the right.



(b) Paths taken by the Warthog robot. Nominal trajectory (black) represents the straight path between the origin and the goal.

Figure 3.9: UGV Warthog Hill-Base-Right experiment description and paths taken.

(a) UGV-Forest-Fork experiment, where the goal is straight ahead.



(b) Paths taken by the Warthog robot. Nominal trajectory (black) represents the straight path between the origin and the goal.

Figure 3.10: UGV Warthog Forest-Fork experiment description and paths taken.

Chapter 4

Conclusions

4.1 Summary

In this work we present a costmap prediction system that predicts what the interactions between the ground and the robot feel like based on environmental characteristics and robot dynamics. To achieve this, we train a network that combines exteroception and information about the robot's velocity to predict a traversability cost derived directly from robot proprioception. We demonstrate that our costmaps enable for more nuanced and diverse navigation behaviors compared to a common baseline. With this work, we hope to push the community towards methods that rely less on human-annotated dataset, and, instead, make use of the multiple sensory modalities in off-road vehicles and temporal correlations to estimate traversability.

4.2 Future Work

One drawback of our method is that it relies on good BEV mapping, which is very sensitive to the ability of the stereo matching algorithm used to find correspondences. One potential future line of work is to enable our method to also make use of front-facing images for prediction, since these front-facing images are more reliable and have longer range. Another weakness is that our predictions are made using patches of terrain rather than the whole BEV map, which could be improved by predicting

a costmap directly from the BEV local map. Finally, our method currently only predicts traversability based on bumpiness, but other similar formulations could be implemented for traversability in terms of slip, or modeling error.

Future work includes online adaptation for better generalization, uncertainty estimation, as well as improved representations for exteroceptive data to overcome our current perception limitations. One key aspect of using self-supervision or cross-modal supervision is that this opens the door to learn models online, since they do not rely on human annotated data. One interesting direction is to study the sample complexity needed to learn local models that perhaps overfit to their current environment but can keep adapting to any environment that the robot encounters. Finally, we have begun work on collecting a larger dataset for off-road navigation to include more modalities, such as lidar and voice annotations, as well as improved infrastructure for training models with this dataset [51].

4.3 Impact

As mentioned at the beginning of this thesis, off-road driving, and more generally, outdoor autonomy has the potential of improving the living conditions of large amounts of the world population. It will also have significant impact in specific applications such as agriculture and humanitarian operations. At the same time, much of this technology has been historically used for defense applications. While this is a significant funding source and consumer of this technology, we hope that the people building upon this work consider applications that benefit humanity over destructive technologies.

Appendix A

Appendix

A.1 Traversability Cost Analysis

To decide the best frequency band for our traversability cost, we collect an evaluation set of 220 5-second trajectories on our robot, spanning smooth, bumpy, sloped, and grassy trails at different speeds. Each of these trajectories is annotated by three human labelers with a score of 1-5 as to how traversable they were, with 5 meaning most difficult to traverse. We calculate the correlation between the average human score and different frequency ranges to tune the frequency range used in our cost function. We found the best frequency range to be 1-30 Hz, which had a Pearson correlation coefficient of 0.66, as shown in Figure A.1. Finally, we use the evaluation set to obtain statistics that we use to normalize the traversability cost function between 0 and 1.

A.2 Mapping

Our method relies on having a dense, colorized point cloud from which we can extract corresponding color and height information about the environment. We experiment with two different setups. When the robot contains a lidar which produces dense point clouds, we can simply colorize these points using the RGB information from a calibrated monocular camera. When the robot is not outfitted with a lidar, or the lidar

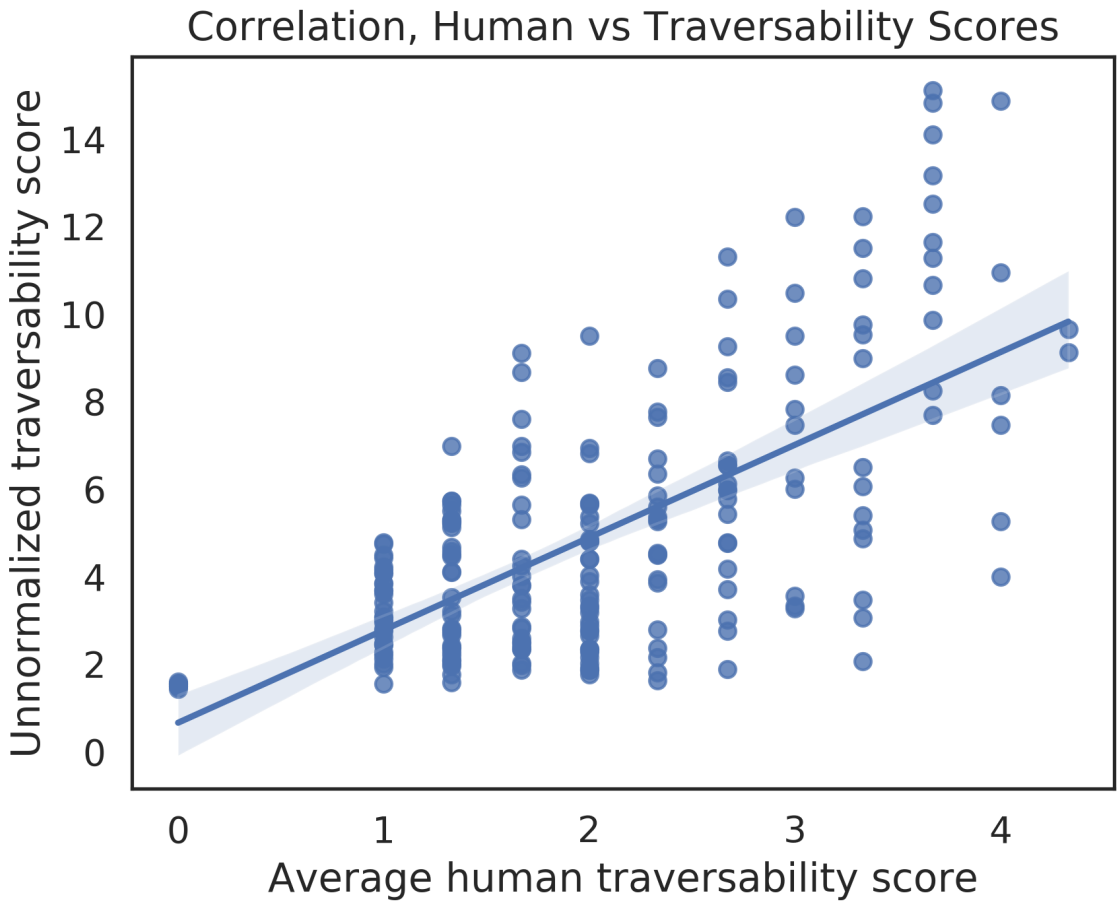


Figure A.1: Correlation between the average of three human-labeled traversability scores (1-5) and our IMU traversability cost function. There is a strong correlation between the two scores, with a Pearson correlation coefficient of 0.66.

produces only sparse point clouds in the near-range, we use a stereo matching network [9] to obtain a disparirty image, from which we estimate the camera odometry using TartanVO [62], a learning-based visual odometry algorithm. We use this odometry and the RGB data to register and colorize a dense point cloud which we then project into a top-down view.

In our setup, we use a 12×12 meter local map with a resolution of 0.02 meters. This results in a 600×600 cell local map consisting of eight channels: three channels for RGB information, four channels containing the minimum, maximum, mean, and standard deviation of the height of the points in each cell, and a channel describing

Algorithm 1 Kinematic Bicycle Model

Current state $s = [x, y, \theta, v, \delta]$ (position, orientation, velocity and steering angle), Control $a = [v_{des}, \delta_{des}]$ (velocity and steering setpoints), Hyperparameters L (wheelbase), K_v (velocity gain), K_δ (steer angle gain) Time derivative of state $\dot{s} = f(s, a)$

$$\dot{s} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} vcos(\theta) \\ vsin(\theta) \\ \frac{vtan(\delta)}{L} \\ K_v(v_{des} - v) \\ K_\delta(\delta_{des} - \delta) \end{bmatrix} \tag{A.1}$$

whether each cell in the local map is unknown. The dimensions of the local map match the dimensions of the learned costmap that will be used directly for path planning.

A.3 Robot Platform

We perform experiments on two different ground robots: a large, Yamaha Viking side-by-side all-terrain vehicle (ATV) modified for autonomous driving by Mai et al. [36], and a Clearpath Robotics Warthog unmanned ground vehicle (UGV). The ATV contains a front-facing Carnegie Robotics Multisense S21 stereo camera, a Velodyne Ultra Puck lidar, a NovAtel PROPAK-V3-RT2i GNSS unit providing IMU data and global pose estimates, as well as an onboard computer with an NVIDIA Geforce RTX3080 Laptop GPU. The Warthog UGV has two FLIR Blackfly S cameras providing a stereo pair with an approximately 53cm baseline, an Ouster OS1-64 LiDAR, a Microstrain 3DM-GX5-35 IMU which provides linear acceleration data and robot odometry measurements, and two Neousys Nuvo 7166GC onboard computers, each with an Intel i9 CPU and NVidia Tesla T4 GPU.

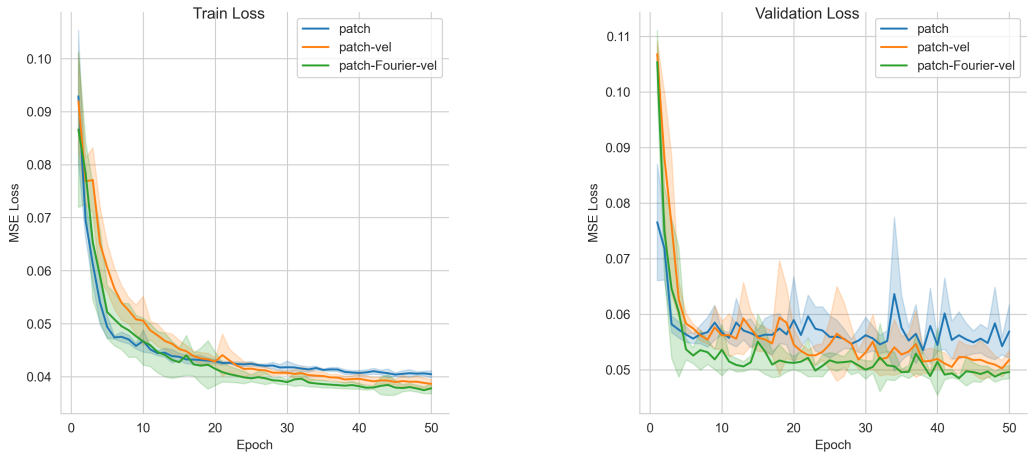
A.4 Navigation Stack

The kinematic bicycle model for the ATV vehicle and the Warthog UGV is shown in Algorithm 1.

Model	Train Loss ($\times 10^{-2}$)	Val. Loss ($\times 10^{-2}$)
Random	22.0 ± 0.14	22.0 ± 0.14
Patch	4.0 ± 0.059	5.7 ± 0.43
Patch-vel	3.9 ± 0.074	5.2 ± 0.15
Patch-Fourier-vel	$\mathbf{3.8 \pm 0.09}$	$\mathbf{5.0 \pm 0.11}$

Table A.1: Training and validation losses for three different models with different inputs. All models were trained with five random seeds. Adding velocity as an input improves training results, and the model which includes Fourier-parameterized velocity performs best (lower is better).

A.5 Training results



(a) Training loss curves.

(b) Validation loss curves.

Figure A.2: Training and validation curves for the different model variations. Including velocity as an input improves both training and validation loss, and including Fourier-parameterized velocity achieves the best results.

We compared three models with different inputs: `patch`, `patch-vel`, and `patch-Fourier-vel`. All of these models used the Torchvision [37] implementation of ResNet18 [20], and were trained using the Adam optimizer [27]. The results are shown in Table A.1. The training hyperparameters for all three models are specified in Table A.2.

We performed an ablation on the inputs to the network to verify whether adding height information improved performance over using just RGB data. As shown in

Hyperparameter	Value	Model
Epochs	50	all
Learning Rate	3×10^{-4}	all
γ (Learning Rate Decay)	0.99	all
MLP Num. Layers	3	all
MLP Num. Units	512	all
CNN Embedding Size	512	all
m (Number of Frequencies)	16	`patch-Fourier-vel`
σ (Frequency Scale)	10	`patch-Fourier-vel`

Table A.2: Hyperparameters used in training three models: `patch`, `patch-vel`, and `patch-Fourier-vel`.

Model	Train Loss ($\times 10^{-2}$)	Val. Loss ($\times 10^{-2}$)
RGB-Fourier-vel	4.0 ± 0.083	4.9 ± 0.06
Patch-Fourier-vel	$\mathbf{3.8 \pm 0.09}$	5.0 ± 0.11

Table A.3: Ablation over model inputs. RGB-Fourier-vel uses only the RGB local map information, whereas Patch-Fourier-vel uses RGB and height statistics extracted from the point cloud. Using both RGB and height information results in slightly better performance in the training set, but comparable performance in the test set.

Table A.3, adding height statistics results in significant improvement at training time, but similar performance at test time. We performed an additional ablation over the parameters for our Fourier frequency parameterization and found that varying the scale of Fourier frequencies does not change performance significantly, and neither did varying the number of sampled frequencies m, as reported in Table A.4.

A.6 High-Resolution Examples

In this section, we show two high-resolution examples comparing the top-down RGB map and the predicted costmap side-by-side, in Figures A.3 and A.4.

Hyperparameter	Value	Train Loss ($\times 10^{-2}$)	Val. Loss ($\times 10^{-2}$)
σ	1	3.8 ± 0.09	4.9 ± 0.06
σ	10	3.8 ± 0.09	5.0 ± 0.11
σ	100	3.7 ± 0.09	5.2 ± 0.17
m	8	3.8 ± 0.04	5.0 ± 0.24
m	16	3.8 ± 0.09	5.0 ± 0.11
m	32	3.8 ± 0.06	5.2 ± 0.37

Table A.4: Ablation over Fourier parameterization parameters. Each hyperparameter choice was evaluated for 5 random seeds. Changing the scale of either the sampled frequencies σ or the number of frequencies samlpled m did not significantly affect performance.

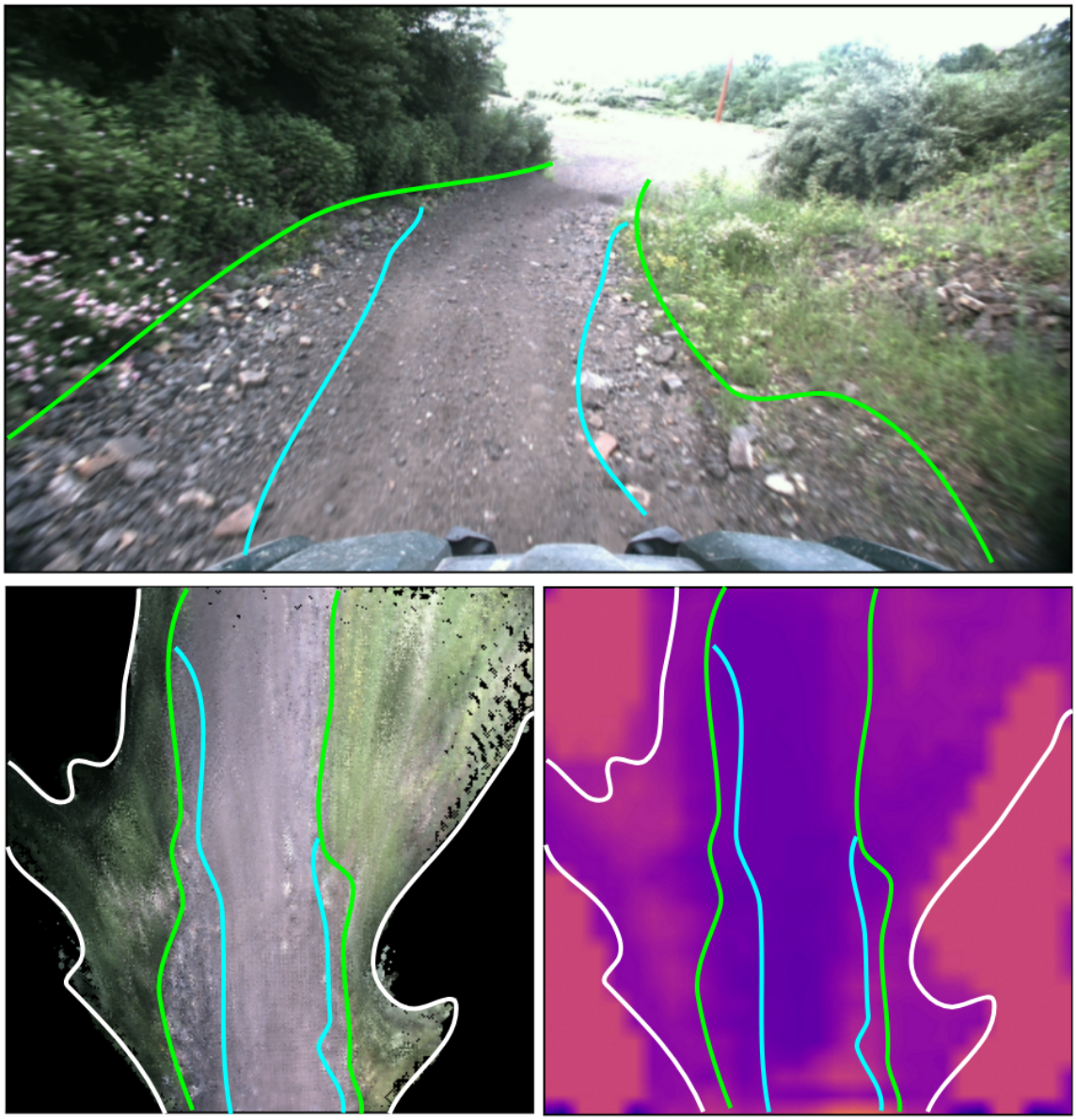


Figure A.3: Side-by-side comparison of front-facing view (top row), RGB map (bottom left) and our predicted costmap (bottom right), with approximate hand-annotations separating the different types of terrain. The white lines enclose the invalid regions, the green lines enclose the regions with vegetation, and the blue lines enclose areas with gravel in the terrain. The predicted costmap predicts a higher cost for grass *and* gravel. Notice that gravel appears as a higher-frequency texture in the RGB map.

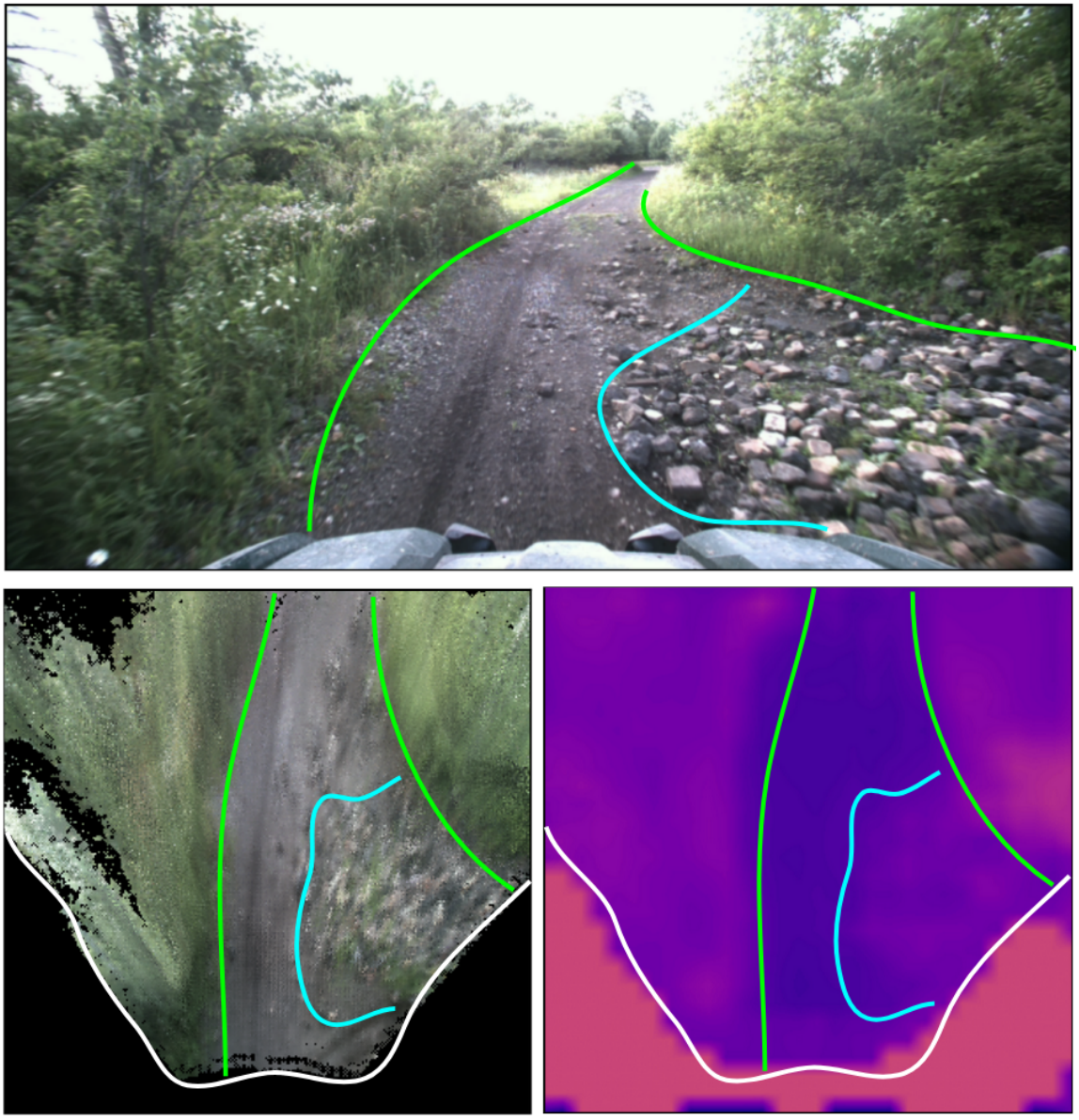


Figure A.4: Side-by-side comparison of front-facing view (top row), RGB map (bottom left) and our predicted costmap (bottom right), with approximate hand-annotations separating the different types of terrain. The white lines enclose the invalid regions, the green lines enclose the regions with vegetation, and the blue lines enclose areas with gravel in the terrain. The predicted costmap predicts a higher cost for grass *and* gravel. Notice that gravel appears as a higher-frequency texture in the RGB map.

Bibliography

[1] Warthog unmanned ground vehicle robot - clearpath, Jan 2021. URL https://clearpathrobotics.com/warthog-unmanned-ground-vehicle-robot/. 3.4.3

[2] Arl autonomy stack, 2022. URL https://www.arl.army.mil/business/collaborative-alliances/current-cras/sara-cra/sara-overview/. Accessed 13-September-2022. 3.4.4

[3] Anelia Angelova, Larry Matthies, Daniel Helmick, and Pietro Perona. Learning and prediction of slip from visual information. *Journal of Field Robotics*, 24(3): 205–231, 2007. 3.2

[4] Max Bajracharya, Andrew Howard, Larry H Matthies, Benyang Tang, and Michael Turmon. Autonomous off-road navigation with end-to-end learning for the LAGR program. *Journal of Field Robotics*, 26(1):3–25, 2009. 3.2

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 3.4.4

[6] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 2.3

[7] Xiaoyi Cai, Michael Everett, Jonathan Fink, and Jonathan P How. Risk-aware off-road navigation via a learned speed distribution map. *arXiv preprint arXiv:2203.13429*, 2022. 3.2

[8] 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. doi: 10.1109/ICRA48891.2023.10160856. 1.2

[9] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In

Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5410–5418, 2018. 3.3.2, A.2

[10] Edmond M Dupont, Carl A Moore, Emmanuel G Collins, and Eric Coyle. Frequency response method for terrain classification in autonomous ground vehicles. *Autonomous Robots*, 24(4):337–347, 2008. 3.2, 3.3.1

[11] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. 2.1.3

[12] 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, 2021. 3.2

[13] David D Fan, Kyohei Otsu, Yuki Kubo, Anushri Dixit, Joel Burdick, and Ali-Akbar Agha-Mohammadi. Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation. *arXiv preprint arXiv:2103.02828*, 2021. 3.1

[14] Péter 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. IEEE, 2018. **??**, **??**, 3.4.4

[15] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1): 23–33, 1997. 2.1.3

[16] Zipeng Fu, Ashish Kumar, Ananye Agarwal, Haozhi Qi, Jitendra Malik, and Deepak Pathak. Coupling vision and proprioception for navigation of legged robots. *arXiv preprint arXiv:2112.02094*, 2021. 3.1

[17] Donald B Gennery. Traversability analysis and path planning for a planetary rover. *Autonomous Robots*, 6(2):131–146, 1999. 3.1

[18] Jason Gregory, Jonathan Fink, Ethan Stump, Jeffrey Twigg, John Rogers, David Baran, Nicholas Fung, and Stuart Young. Application of multi-robot systems to disaster-relief scenarios with limited communication. In *Field and Service Robotics*, pages 639–653. Springer, 2016. 3.4.2

[19] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009. 3.2

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3.3.4, A.5

[21] Andrew Howard, Michael Turmon, Larry Matthies, Benyang Tang, Anelia

Angelova, and Eric Mjolsness. Towards learned traversability for robot navigation: From underfoot to the far field. *Journal of Field Robotics*, 23(11-12):1005–1017, 2006. 3.2

[22] Larry D Jackel, Eric Krotkov, Michael Perschbacher, Jim Pippine, and Chad Sullivan. The darpa lagr program: Goals, challenges, methodology, and phase i results. *Journal of Field robotics*, 23(11-12):945–973, 2006. 3.2

[23] Chiyu Max Jiang, Andre Cornman, Cheolho Park, Ben Sapp, Yin Zhou, and Dragomir Anguelov. Motiondiffuser: Controllable multi-agent motion prediction using diffusion, 2023. 2.3

[24] Peng Jiang, Philip Osteen, Maggie Wigness, and Srikanth Saripalli. Rellis-3d dataset: Data, benchmarks and analysis. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 1110–1116. IEEE, 2021. 2.3

[25] Gregory Kahn, Pieter Abbeel, and Sergey Levine. Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, 6(2):1312–1319, 2021. 3.1, 3.2

[26] Dong-Ki Kim, Daniel Maturana, Masashi Uenoyama, and Sebastian Scherer. Season-invariant semantic segmentation with a deep multimodal network. In *Field and service robotics*, pages 255–270. Springer, 2018. 3.1

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3.3.4, A.5

[28] 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. *arXiv preprint arXiv:2211.12732*, 2022. 2.3

[29] Kurt Konolige, Motilal Agrawal, Morten Rufus Blas, Robert C Bolles, Brian Gerkey, Joan Sola, and Aravind Sundaresan. Mapping, navigation, and learning for off-road traversal. *Journal of Field Robotics*, 26(1):88–113, 2009. 3.2

[30] Philipp Krüsi, Paul Furgale, Michael Bosse, and Roland Siegwart. Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments. *Journal of Field Robotics*, 34(5):940–984, 2017. 3.1, 3.2, 3.4.4

[31] Jean-François Lalonde, Nicolas Vandapel, Daniel F Huber, and Martial Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of field robotics*, 23(10):839–861, 2006. 3.1, 3.2

[32] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European conference*

on computer vision, pages 1–18. Springer, 2022. 2.3

[33] Tingting Liang, Hongwei Xie, Kaicheng Yu, Zhongyu Xia, Zhiwei Lin, Yongtao Wang, Tao Tang, Bing Wang, and Zhi Tang. Bevfusion: A simple and robust lidar-camera fusion framework. *Advances in Neural Information Processing Systems*, 35:10421–10434, 2022. 2.3

[34] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. In *European Conference on Computer Vision*, pages 531–548. Springer, 2022. 2.3

[35] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 3.2

[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. A.3

[37] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010. A.5

[38] Daniel Maturana, Po-Wei Chou, Masashi Uenoyama, and Sebastian Scherer. Real-time semantic mapping for autonomous off-road navigation. In *Field and Service Robotics*, pages 335–350. Springer, 2018. 2.1.3, 3.1, 3.2

[39] Johan R Meijer, Mark AJ Huijbregts, Kees CGJ Schotten, and Aafke M Schipper. Global patterns of current and future road infrastructure. *Environmental Research Letters*, 13(6):064006, 2018. 1.1

[40] Xiangyun Meng, Nathan Hatch, Alexander Lambert, Anqi Li, Nolan Wagener, Matthew Schmittle, JoonHo Lee, Wentao Yuan, Zoey Chen, Samuel Deng, et al. Terrainnet: Visual modeling of complex terrain for high-speed, off-road navigation. *arXiv preprint arXiv:2303.15771*, 2023. 2.1.4

[41] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 3.1

[42] Jiquan Ngiam, Vijay Vasudevan, Benjamin Caine, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In *International Conference on Learning Representations*, 2021. 2.3

[43] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network.

Advances in neural information processing systems, 1, 1988. 2.2.1

[44] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019. 3.3.3

[45] Roy Richardson. Dirt, gravel amp; low volume road maintenance program. URL https://www.agriculture.pa.gov/Plants_Land_Water/StateConservationCommission/DGRMP/Pages/default.aspx. 1.1

[46] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 2.2.1

[47] Adarsh Jagan Sathyamoorthy, Kasun Weerakoon, Tianrui Guan, Jing Liang, and Dinesh Manocha. Terrapn: Unstructured terrain navigation using online self-supervised learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7197–7204. IEEE, 2022. 3.2

[48] Amirreza Shaban, Xiangyun Meng, JoonHo Lee, Byron Boots, and Dieter Fox. Semantic terrain classification for off-road autonomous driving. In *Conference on Robot Learning*, pages 619–629. PMLR, 2022. 3.2

[49] Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models. In *Conference on Robot Learning*, pages 674–684. PMLR, 2022. 3.2

[50] 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)*, pages 4844–4850. IEEE, 2021. 1.2, 3.2

[51] 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. 2.3, 4.2

[52] Jonathan Spencer, Sanjiban Choudhury, Matthew Barnes, Matthew Schmittle, Mung Chiang, Peter Ramadge, and Sidd Srinivasa. Expert intervention learning: An online framework for robot learning from explicit and implicit human feedback. *Autonomous Robots*, pages 1–15, 2022. 2.2.1

[53] David Stavens and Sebastian Thrun. A self-supervised terrain roughness estimator for off-road autonomous driving. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 469–476, 2006. 3.2, 3.3.1, 3.3.3

[54] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 2.3

[55] Wen Sun, Arun Venkatraman, Geoffrey J. Gordon, Byron Boots, and J. Andrew Bagnell. Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3309–3318. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/sun17d.html. 2.2.1

[56] Gokul Swamy, Sanjiban Choudhury, J Andrew Bagnell, and Steven Wu. Of moments and matching: A game-theoretic framework for closing the imitation gap. In *International Conference on Machine Learning*, pages 10022–10032. PMLR, 2021. 2.2.1

[57] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020. 3.1, 3.3.3

[58] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006. 3.2

[59] 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. IEEE, 2022. 2.3, 3.2, 3.4.1

[60] 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. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 924–930, 2023. doi: 10.1109/ICRA48891.2023.10161268. 1.2, 2.1.4, 2.2.1, 3.2

[61] Gabriel Günter Waibel, Tobias Löw, Mathieu Nass, David Howard, Tirthankar Bandyopadhyay, and Paulo Vinicius Koerich Borges. How rough is the path? terrain traversability estimation for local and global path planning. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):16462–16473, 2022. 3.2

[62] Wenshan Wang, Yaoyu Hu, and Sebastian Scherer. Tartanvo: A generalizable learning-based vo. *arXiv preprint arXiv:2011.00359*, 2020. 3.3.2, 3.4.1, A.2

[63] Peter 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. 3.3.1

[64] 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. 3.1, 3.2, 3.3.1

[65] 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 *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5000–5007. IEEE, 2019. 2.3

[66] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017. 2.1.5, 3.2, 3.4.2

[67] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*, 2023. 2.3

[68] 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. IEEE, 2022. 3.2

[69] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021. 2.3

[70] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020. 1.1

[71] 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. 2.1.2, 3.4.2