Off-Road Navigation Under Sensing Uncertainty

Matthew Schmittle

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2025

Reading Committee:
Siddhartha Srinivasa, Chair
Byron Emereth Boots
Sanjiban Choudhury

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Off-Road Navigation Under Sensing Uncertainty

Matthew Schmittle

Chair of the Supervisory Committee:
Siddhartha Srinivasa
Computer Science and Engineering

Off-road autonomous robots navigate through unstructured environments using onboard computation and sensing without prior maps or adherence to conventional traffic rules. This challenges these systems to efficiently determine where they can and can't drive on the fly. Diverse testing environments and incomplete sensing due to occlusions or limited sensor range make it difficult to accurately understand the environment. A motion planner in this context focuses on predicting the robot's actions seconds to minutes ahead, considering environmental obstacles, vehicle dynamics, and unknown regions. Existing systems that ignore uncertain or incomplete sensing are prone to dangerous decisions from over-optimism or to unnecessarily inefficient paths from over-pessimism. This thesis aims to address those shortcomings by explicitly reasoning about sensing uncertainty. This dissertation proposes three strategies for tractably reasoning over and reducing uncertainty in the context of off-road autonomy. We discuss the pitfalls of not reasoning about uncertainty and present a simple local uncertainty-aware algorithm that traverses safer, more efficient paths. Then we consider uncertainty outside the local map and leverage sparse long-range sensor information to guide planning. We show this algorithm makes less myopic earlier decisions to avoid distant obstacles. Finally, to reduce uncertainty, we introduce a technique that harnesses off-the-shelf video footage to rapidly adapt both perception and planning for new environments, achieving performance comparable to that obtained from costly, hard-to-collect robotic data. The research is grounded heavily in real-world applicability with the objective of minimizing time to reach the goal and number of safety interventions. Conducted during the participation in the DARPA RACER program, this research addresses the real-world challenges that uncertainty posed for the robots.

Acknowledgments

This dissertation would not be possible without the many people that supported me along the way. Thank you to my advisor, Sidd Srinivasa, for pushing me to be my best and constantly raising the bar. Whether it was a new algorithm or tweaking a particular figure, your high standard for doing good work mixed with positive encouragement motivated me to jump higher and reach further. In particular, I greatly appreciate you adding me onto both MuSHR and RACER, two projects that shaped my experience and gave me the learning experience I needed.

Thank you to my thesis committee members Byron Boots, Sanjiban Choudhury, and Sawyer Fuller. Byron, RACER has certainly been both a literal and metaphorical wild ride. I thank you for your RACER team leadership, your constant frustration with the state of autonomy pushing us to be better, and most importantly your patience with all the "Schmittle loops." Sanjiban, while it has been many years since we worked together at the start of this Ph.D., I will never forget the lessons you taught me. I am immensely grateful for you taking me under your wing that first year.

Thank you to DARPA and CRL, in particular Stewart Young, Scott Fish, Doug Hackett, and Mike Perschbacher. You all created a program I was not ready for, but was exactly what I needed. The design and logistics, focus on scientific insights, and constant push made an environment that let UW explore the cutting edge in a meaningful way. Without the RACER program I don't know what this thesis would be about. Also thank you to DARPA for funding the RACER program, I hope more like it can be created in the future.

While Ph.D. programs can feel like a long marathon, it helps to have others by your side to keep you moving. To the MuSHR team: Johan, Matthew, Colin, and Patrick, Sidharth, I found your enthusiasm and team spirit inspiring and I am still so impressed by what we accomplished. To the RACER team, I can't name everyone without this section being immense, but I appreciate you all for your kindness, grit, and constant reminder that the planner could be better. Jakub, Nathan, Grady, Nolan, Jon, Amir, Xiangyun, Sammy, Tim, Tyler, Patrick, and Sanghun, I am grateful for all that you taught me and I couldn't picture a software team I would rather sit in a shipping container for twelve hours a day with. Roger and Eliot, thank you for fixing the robot every time we crashed it; you two are really the rockstars of the team and I always loved how you guys could put the robot back together faster than we could fix the bug. And Greg, whether it's driving to Ellensburg or riding shotgun in an autonomous robot, thanks for all the car chats and perspective, but never forget – Modelo won. Thank you

to all my labmates over the years: Sherdil, Tapo, Ethan, Amal, Brian, Gilwoo, Aditya, Christophoros, Kay, Helen, Harine, Mateo, and Yunchu for sharing in this experience and supporting one another. And Khimya Khetarpal, thank you for your guidance and support during my final year–your advice on research and grad school made a big difference.

To the my friends both in work and in life, Rosario, Adam, Alex, and Selest, I can't thank you enough for being the sounding board for any crisis or big decision I was facing. I can only hope I provided at least half as much support you all provided me through tough times. And thank you to Jenny, Jenai, and Youmna for not letting us talk about work too much.

This thesis would really not be possible without the immeasurable support of Rohan "Beachball" Baijal. He was not only a co-author on every major paper, but his willingness to field test in freezing temperatures late at night, his constant questioning forcing me to strengthen my reasoning, and his incredible kindness made him a wonderful person to work with. Rohan, thank you.

Thank you to Sabine Grupp (LG), your love and support make every day shine brighter. Thank you for your patience with this program and with me. It's not easy watching the one you love struggle, but I'm grateful you always have my back. And thank you to Hank, for clawing me every morning to get up and start my day.

To my friends, Carrie, Jamie, Ron, Paige, Connor and many more, thank you for bringing me outside of my school bubble and helping me see the world from a different perspective.

Finally, thank you to the Fam Bam for your unwavering support despite me moving 3000 miles away to pursue this degree. You are the coolest people I know and I am inspired to make you proud every day.

Contents

List of Figures

List of Tables

1
Introduction

While there has been a well-known evolution in on-road autonomous cars [111; 104; 112] now driving passengers in multiple cities, there is also a quiet revolution in lesser-known off-road autonomy. Off-road autonomy (a part of Field Robotics), as the name implies, is when autonomous vehicles or robots operate in unstructured natural environments where there are fewer rules to follow and more reliance on local perception and planning. Off-road autonomy has been an active area of research for decades. It started in 1989 with ALVINN [75], a van autonomously driving down a dirt road. Then, it grew to the 2005 DARPA Grand Challenge [21] where Stanley drove 212 kilometers through desert trails. And in more recent years the RACER program [20] which have driven tens of kilometers in truly off-road terrain(for full history see Chapter 2). But why is off-road autonomy important? As one can expect by DARPA's involvement, there are military applications where robots transporting goods, collecting reconnaissance, and generally removing people from warfare are important. Beyond that, off-road autonomy is useful for a number of applications, notably, mining, search and rescue, construction, environmental monitoring, and wildfire fighting. These applications all require an autonomy system capable of dealing with less-structured, ever-changing environments, and lots of uncertainty. This thesis looks specifically at perception and planning's role in handling uncertainty.

Unlike on-road driving (which contends with lanes, signs, and rules of the road), off-road driving has much less structure. The robot can go wherever it can effectively traverse. Further, with no prior information such as maps of the environment, the robot is limited only to what it can perceive with its onboard sensors. So perception and planning must determine online what it can and cannot traverse and how it will traverse it. For example, an on-road car knows the drivable road will continue around the blind corner, whereas in off-road you don't know what could lie around the turn. Compounding on all of this, the terrain is simply much rougher than engineered roads, increasing sensor noise and the need for precise decision making. One notable relaxation in off-road autonomy is safety. Off-road robots generally operate in less-populated areas where the risk to human life is lower and a human generally is not riding in the off-road robot. This can simplify the problem, but more importantly, enables autonomy to push its limits and perform tasks a human would not, for example, underground mine inspection in high-risk mines [88].

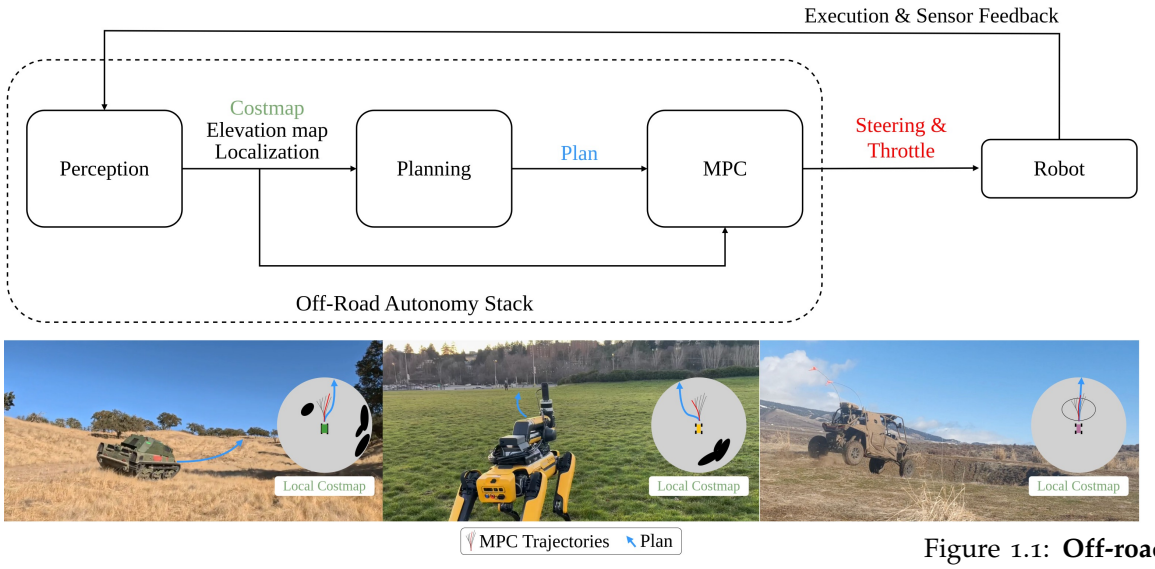


Figure 1.1: **Off-road Autonomy Stack.** Perception ingests sensor data and outputs maps and localization information. Planning ingests maps and outputs plans (BLUE). Model predictive control (MPC) outputs steering and throttle commands as part of the MPC trajectories.

This thesis was completed as part of, and funded by, the DARPA Robotic Autonomy in Complex Environments with Resiliency (RACER) program [20]. RACER challenges teams to program an autonomous full-size off-road vehicle equipped only with on-board sensing and compute to navigate complex terrain (deserts, forests, hills) over long distances (10s of km). The task is to travel from start to goal as fast as possible while minimizing safety interventions. A safety intervention is whenever a human operator or low-level system must intervene because the robot is executing behavior deemed dangerous to itself. Notably, the robots are given no prior maps of the environment and must navigate based purely on their local sensors. We focus on this specific setting as part of this thesis.

Modern off-road autonomy stacks (Fig. 1.1) follow a general hierarchy of perception, planning, and control [107; 73]. Perception ingests sensor data and produces a 2D/3D map of the local environment - segmenting obstacles and free space. The motion planner ingests the map and finds a path from the robot to the goal considering the full horizon of the map. Control/local planning ingests the plan and the map and then follow that path considering short term control costs such as rollover risk. This is all done on-board the robot in real time and is an impressive feat considering the robot moves between 5 and 10 m/s.

Further from the robot, there are fewer and noisier sensor measurements from the onboard sensors making perception more challenging. Perception can be unsure of the presence (accuracy) and/or the exact location (precision) of obstacles. Parts of the environment receive no sensor readings at all due to occlusions. Inpainting, filling in holes based on surrounding map context, has risen as a promising approach to addressing these occlusions [86; 60]. But eventually, far enough out, there is insufficient sensing to produce a dense map of the environment and practitioners generally ignore further sensor readings maintaining only a "local map" [60; 25; 73]. The space beyond the local map is deemed entirely unknown. These forms of perception uncertainty, if not considered, can cause problems for downstream planning. For example, giving

the planner the map with the most likely costs can have both false free space and false obstacles that the planner is entirely unaware of and may plan through. The planner can do little with unknown space beyond the map, often resulting in a straight-line path to the goal.

This research examines uncertainty inside and beyond the local map, aiming to create a **real-time motion planner that robustly navigates the robot through the 'fog of uncertainty.'** This research is grounded heavily in real-world applicability, including studies on real-world robots with the goal of minimizing time to reach the goal (traversal time) and number of safety interventions.
More specifically, this thesis is guided by the following research question:

RQ-Thesis *How can we develop perception and planning algorithms to tractably navigate under the fog of uncertainty?*

To answer RQ-Thesis, this thesis presents works that investigate the following research questions:

RQ1 What is the effect of mismatch between the determinized world and the true world?
RQ2 How can we combat the effects of determinization mismatch?
RQ3 Using only on board sensing, is it possible to traverse a similar path as one planned using privileged global information?
RQ4 Where is long-range navigation not useful?
RQ5 Can off-robot experience improve decision making under uncertainty?

In pursuit of answering RQ-Thesis, the following *key challenges* exist for planning algorithms on off-road autonomous robots.

- Off-road environments are riddled with uncertainty. It can be both difficult to know accurately what something is (e.g., bush or rock) and where precisely it is.
- Uncertainty from perception will negatively affect planning when incorrect predictions occur.
- Planning is limited by the mapping range. If the planner reasons only up to its maximum map range, but additional sensor information exists beyond, it is inherently myopic.
- Real-time performance is required for practical deployment of any approach.
- With each new environment, perception will need to adapt quickly to minimize its uncertainty.

Our *key observation* is that perceiving everything perfectly is both impossible and unnecessary for planning to make effective decisions. We need to think beyond optimizing for pixel-wise segmentations and ask what the planner really needs from perception. This leads to our **key insight**, which guides the work in this dissertation:

To reason effectively about uncertainty in off-road navigation, perception should focus less on just mapping the world, but rather perceiving what is relevant to planning.

Given this, the dissertation will focus on three primary research thrusts: (1) tractable planning under sensing uncertainty; (2) leveraging sparse long-range information to guide planning; and (3) quickly adapting to new environments.

1.1 Background

This section will cover relevant background in off-road autonomy and approaches for handling uncertainty. In addition, Chapters 3, 4, 5 will all go into more in depth on related works specific to the techniques of the chapter.

As mentioned previously, modern off-road autonomy stacks follow the perception, planning, and control paradigm shown in Fig. 1.1. We cover each component below with a focus on uncertainty reasoning. Table 1.1 provides a categorization of uncertainty-specific works described along six categories: perception focused, planning focused, fast adaptation in new environments, reasoning about unknown space (non-mapped), reasoning about mapped perception uncertainty, and system dynamics uncertainty.

1.1.1 Uncertainty-Aware Perception

The role of perception is to build a compact representation of the environment from sensor data that can be ingested by planning. In off-road autonomy, this generally manifests as a 2D bird's-eye-view map [25]. The map contains traversable and non-traversable regions as either a scalar cost per cell or binary occupancy. Early works relied on both robot sensing data and satellite imagery to populate the maps [90; 93; 91]. Additionally, early works on robot platforms equipped with only stereo cameras explored polar map representations where where cells form angular bins rather than a regular grid [7; 85]. This allowed for the map resolution to naturally capture stereo error with increased distance. This representation was combined with near to far learning [27; 62; 63; 7; 108; 129]. It relies on two key observations: cameras provide long-range sensor information, and sensing near the robot is reasonably accurate. As such, these approaches train an image classifier to distinguish traversable and non-traversable terrain. Training is self-supervised by using accurate near-range sensing to provide labels for previous timesteps when the observed region was far away from the robot. Once trained, traversability was projected from image space to the polar map for planning.

More recently, to fill unknown regions of the map with no sensor readings, inpainting [86; 60; 28] has become a common approach. Given surrounding map context, inpainting networks predict what is likely in the unknown space. This works by training an inpainting network from a dataset of instantaneous sensor readings (incomplete) paired with labeled complete maps created from running SLAM over many timesteps.

Self-supervised traversability estimation has been a well-studied problem in off-road autonomy [54; 115; 118; 45; 80; 46; 92]. By being self-supervised, it forgoes manual

labeling enabling faster adaptation in new environments – reducing uncertainty. The model either predicts traversability from 3D lidar data or 2D image data. The self-supervision signal is a combination of both where the robot was manually driven and proprioceptive sensing to parse relevant data like terrain roughness and energy expenditure. From aligning this information with sensor readings of the driven path, we can learn a mapping from sensors to traversability, roughness, and other quantities. Most notably, recent works [32; 92] are able to train traversability models online or with only seconds of robot data.

1.1.2 *Uncertainty-Aware Planning*

The role of planning is to produce feasible low-cost plans through the map provided by perception. There has been a fair amount of prior work on dynamic re-planning under uncertainty. D* and D* Lite are well-known dynamic re-planning search algorithms that have been demonstrated to quickly re-plan in real-world settings [97; 49; 29]. They both re-use the search tree to rapidly re-plan when the environment is perceived to change. Neither is designed to reason about uncertainty directly, but rather react quickly to new changes in the environment. Field D* [29] in particular was a practical adaptation of D* that worked on real-world off-road robots by creating smooth trajectories via interpolation.

Another common planning approach is the adaptive state lattice (ASL) [39; 37]. A state lattice [74] is a planning data structure that is formed by combining control sets and enables planning under kinematic constraints and efficient graph search. ASLs go a step further and optimize the lattice based on the current map to make planning faster and produce better paths. EASL [37] then makes ASLs tractable for real time applications by discretizing the set of possible per-node adaptations and leveraging previous planning queries. EASL's real-time performance has thus made it an attractive planner for off-road robotics. EASL does not reason about uncertainty directly, but given its prominence in off-road planning, it is worth mentioning.

To directly address planning uncertainty, risk-aware planning has become a popular approach [17; 8; 101; 30; 65; 13]. In particular, Cai et al. [13] learns a speed distribution for each cell in the costmap then computes the Conditional Value at Risk (CVaR) [110] metric to consider the risk of traversing each cell. The risk is directly incorporated as a cost for planning. Though we show in Chapter 3 how viewing risk simply as a cost can be problematic because it ignores the true likelihood of an event.

Less common in off-road planning but nonetheless relevant, is planning over Partially Observable Markov Decision Processes (POMDPs). POMDPs have a plethora of approaches, but a notable few rely on a fixed or sampled set of MDPs to make the problem more tractable [79; 89; 94; 100; 15]. While promising, solving a POMDP is still PSPACE-complete [70] and determinization, making a deterministic approximation of a stochastic problem, has arisen as a tractable technique for relaxing the problem [126]. In particular, a variety of works [38; 4; 99; 123; 17] have used posterior sampling [105] to make the planning problem tractable, requiring only sample access to the posterior. Dynamic replanning with posterior sampling (DRPS) [38] samples one problem and

solves it optimally, letting sampling naturally balance exploration and exploitation. Notably, none of these works has been applied to real-world off-road planning, and Chapter 3 is the first known to do so.

1.1.3 Uncertainty-Aware Model Predictive Control

Model Predictive Control (MPC) [34] is a receding-horizon optimal control strategy. At each time step, it solves a finite-horizon optimization problem to choose the best control sequence to track the path from planning while respecting system dynamics and reacting to local obstacles. Because of its predictive model and finite planning horizon, we also refer to it as local planning. Local planning does not focus as much on uncertainty from perception, but rather on uncertainty in vehicle dynamics. Because the predictive model is only a fast approximation of the true dynamics, re-planning is key to managing this uncertainty. Early work in handling uncertain vehicle dynamics for off-road robots used maneuver dictionaries [84] which store recorded robot trajectories that can then be used instead of potentially inaccurate predictive models. However, they were limited to 2D scenarios. More recently, Model Predictive Path Integral (MPPI) control [122] has shown impressive performance on various off-road vehicles. Because of its sampling procedure, it is not limited to differentiable costs or dynamics, and it is highly parallelizable. This allows MPPI to explore many possible trajectories at once and replan fast to adjust to uncertain dynamics. These properties have cemented MPPI as the go-to approach for local planning in off-road autonomy.

	Perception	**Planning**	**Fast adaptation**	**Unknown space uncertainty**	**Perception uncertainty**	**Dynamics uncertainty**
Near-to-Far Learning [27]	Y	N	Y	Y	Y	N
Costmap Inpainting [60]	Y	N	N	Y	N	N
SS Traversability [92]	Y	N	Y	N	Y	N
Risk-Aware Planning [13]	N	Y	N	N	Y	N
Determinization [38]	N	Y	N	N	Y	N
MPC [122]	N	Y	N	N	N	Y
DREAMS (Chapter 3)	N	Y	N	N		N
LRN (Chapter 4)	Y	Y	N	Y	N	N
Video Tracking (Chapter 5)	Y	Y	Y	N	Y	N

Table 1.1: **Categorization of uncertainty in off-road autonomy.** Cited are the most recent works for brevity.

2
A Brief History of Off-Road Autonomy

This chapter provides a brief history of notable programs that led to the off-road autonomy today. Each program provided funding and structure to facilitate many research papers and real-world deployments. We highlight relevant programs but note that this overview is not comprehensive. For brevity, we have omitted much of the legged outdoor-autonomy development. Fig 2.1 provides a graphical timeline of the programs we mention, culminating in the RACER program.

2.1 ALVINN, the first off-road autonomous vehicle (1989)

The story of off-road autonomy begins with ALVINN (Autonomous Land Vehicle In a Neural Network) [75]. It was a three-layer neural network that steered CMU's NAVLAB van along narrow, 400 m wooded dirt path at 0.5 m/s. The van was unique in that it not only carried onboard compute, which was much larger at the time, but also the researchers themselves! This work set the stage for what was to come, and showed it was possible to autonomously perceive and navigate unpainted trails.

2.2 PerceptOR (2000 - 2003)

The Perception for Off-Road Robotics program (PerceptOR) [51] was the first of its kind off-road autonomy program. It set a new rigorous testing standard for evaluation. Evaluations were performed by the independent government team where the contracted performers were given no advance knowledge of the evaluation courses. Further, robots went off-trails and navigated complex terrain performing navigation over multiple kilometers at a max speed of 0.3 m/s.

2.3 DARPA Grand Challenge (2004-2007)

DARPA's first Grand Challenge [2] in March 2004 set a 228 km course from Barstow, CA to Primm, NV in the Mojave Desert; none of the 15 entrants completed it. Eighteen months later, Stanford's Stanley [106; 21] conquered the revised 212 km route in 6h 53 min by fusing imagery, lidar, and radar to create occupancy maps and a probabilistic road detector, taking the $2M prize. This event is noted as the birth of the

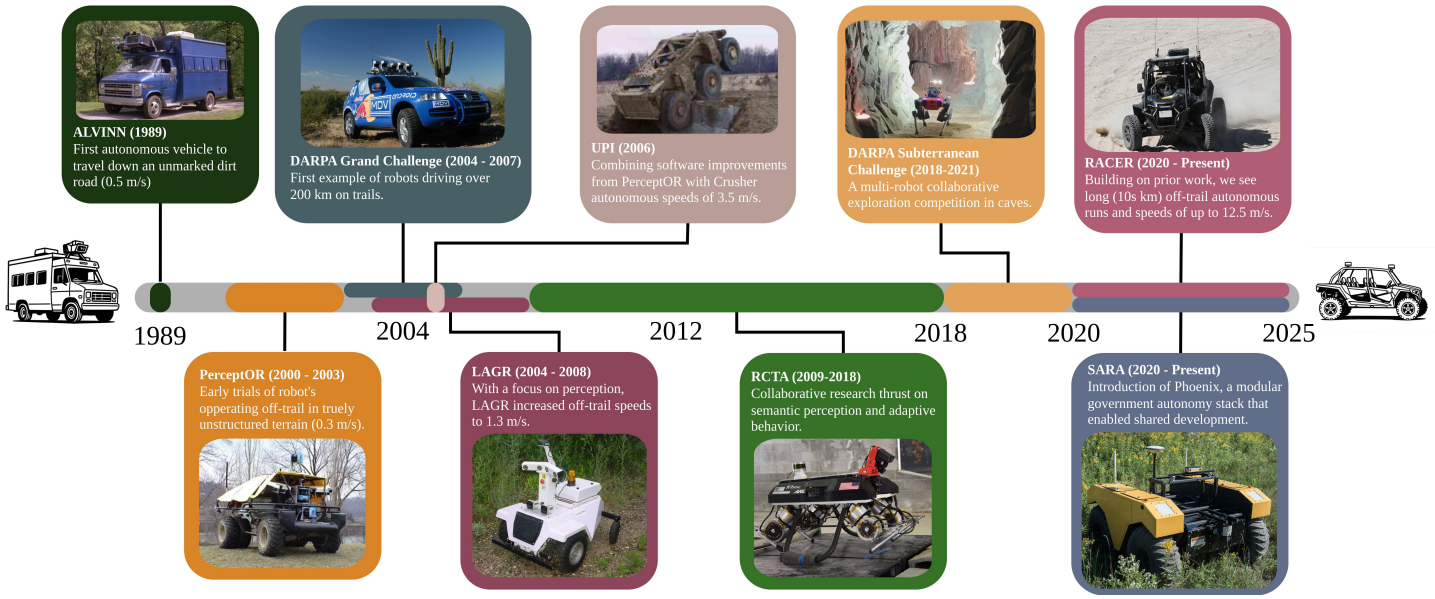


Figure 2.1: **Timeline of Off-road Autonomy.** From 1989 to 2025 this timeline summarizes all major programs that pushed autonomy forward.

autonomous vehicle (AV) boom where many participating researchers went on to start AV companies and participate in the DARPA Urban Challenge [109]. The DARPA Grand Challenge showed robotic vehicles could perform robustly over long distances but still required driving on dirt-trails. Taking robots into spaces with no trails remained an open problem to be addressed by other DARPA projects.

2.4 *LAGR (2004–2008)*

Learning Applied to Ground Robots (LAGR) [43; 42; 40] continued with the PerceptOR's off-trail focus, but emphasized learning-based perception methods. Ten university teams, each given an identical skid-steered robot with stereo cameras, met monthly on forested "time-trial" courses. This platform standardization was a departure from past programs as they found a standard platform allowed performers to focus on autonomy not on hardware design. Notably this program spurred a focus on Near To Far Learning [69; 66; 12; 129; 108; 7; 62; 63; 116], a self-supervised traversability approach where near-range sensors (10 m) provided labels for training far-range perception (50 m). By the end of the program, teams completed 100 m long courses with the vehicle's top speed of 1.3 m/s.

2.5 *UGCV PerceptOR Integrated (UPI) (2006)*

The UGCV PerceptOR Integrated (UPI) program [6] combined advances from the PerceptOR and LAGR programs with advanced hardware of the crusher platform [98]. The combined system was able to traverse a 20 km desert course at an average autonomous speed of 3.5 m/s.

2.6 *Robotics Collaborative Technology Alliance (RCTA) (2009-2018)*

The Army's Robotics Collaborative Technology Alliance (RCTA) [53] pulled ARL, CMU, MIT, GDLS, and others into a decade-long push on semantic perception, adaptive behavior and meta-cognition. The flagship hardware proof point—LLAMA [1], a 75 kg, all-electric quadruped—combined proprioceptive controls with off-road perception to keep pace with soldiers across rubble. RCTA codebases and datasets later seeded ARL's Government-owned Ground Autonomy Stack.

2.7 *DARPA Subterranean Challenge (2018-2021)*

The Subterranean Challenge [68] forced autonomy underground, where teams of robots had to explore cave systems with no GPS and little to no light. In the 2021 Final, Team CERBERUS [107] with a team of quadrupeds and drones mapped a tunnel network under dust and darkness, taking the $2M Systems prize. While different from the high-speed off-road navigation problem, the semantic mapping, multi-robot coordination, and degraded-vision pipelines created new opportunities for off-road ground vehicles and disaster-response robots.

2.8 *Scalable Adaptive and Resilient Autonomy (SARA) (2020 - present)*

SARA [18] reinvigorated focus on off-road autonomy with `Phoenix`, a government-maintained autonomy stack capable of running on multiple ground platforms. University collaborators could then build on the existing stack, swapping or adding components to accelerate development. `Phoenix` served as the foundation that RACER built on.

2.9 *Robotic Autonomy in Complex Environments with Resiliency (RACER) (2020 - present)*

The RACER program [20] combined all the findings from previous programs and recent advancements in perception, planning, and control to push off-road autonomy to its limit. Like LAGR, RACER provided standardized Racer Fleet Vehicles and Racer Heavy platforms to performers. Like PerceptOR, RACER held regular experiments on new unseen terrain where performers were not given prior knowledge of courses. Under this setting, teams were able to push autonomy up to 12 m/s with courses in the tens of kilometers. This thesis was conducted as part of the University of Washington RACER team.

2.10 *Patterns and Prospects*

Across three decades the field has followed a recurring pattern: A DARPA or Army-led initiative challenging teams to perform on real-world courses, an increase in operational speed and robustness, and technology transition into the next program. Researching and field-testing during these programs shows that moving robots out of

controlled labs and into challenging outdoor environments exposes critical gaps and drives autonomy forward. As mentioned by Team CERBERUS' review [107], *"Many of the features and development directions of the CERBERUS autonomy solutions have evolved from our experiences in the field tests."*

While approaches varied, they all roughly followed the perception planning and control pipeline in Fig. 1.1. Field testing reveals that while improving individual components is important, the integration of components is equally, if not more important. This is the motivating idea of this thesis, reconsidering how perception and planning should interact to better handle the challenges of real deployment.

Ongoing research thrusts in off-road autonomy include fast adaptation, traversing negative obstacles, and failure recovery. Recent near to far self-supervised methods can train image-based traversability models in only seconds of experience [32; 92], enabling rapid adaptation to new environments. Negative obstacles—depressions, trenches, and ditches—remain challenging because limited-view LiDAR seldom captures their full geometry and the interaction dynamics are intricate. Recent methods have achieved reliable control across known ditches [36], but general solutions are still nascent. Finally, failure detection and recovery need deeper attention: catastrophic events such as a quadruped tipping over are easy to flag and reset, yet slow-burn failures–e.g., a robot oscillating or aimlessly circling–often evade current monitors. Addressing these subtler behaviors will be essential for truly resilient off-road autonomy.

3
Tractable Planning under Uncertainty

This chapter addresses research questions RQ1 and RQ2, concerning tractable planning under uncertainty. This involves motion planning over an uncertain world while considering multiple likely outcomes, all while maintaining tractability for relevance in a real-time setting. We focus on uncertainty in obstacle locations, which could come from aleatoric or epistemic sources in perception.

3.1 Introduction

As stated previously, an off-road autonomous robot can drive wherever it can effectively traverse. This flexibility requires the onboard sensors and perception system to discern what terrain is and is not traversable, which usually comes in the form of a map, a compressed representation of the environment. Terrain that is far from the robot may be difficult to classify precisely due to natural occlusions (hills, trees), few sensor readings, or lack of training examples in the given environment. Noisy perception creates both false obstacles and false free space in the map; a downstream planning algorithm that is unaware of this uncertainty may produce dangerous collision-bound or roundabout paths. Thus, uncertainty is the core challenge of long-range planning; an algorithm must appropriately consider this sensing uncertainty from perception when planning paths.

In this setting, a robot plans paths through a previously unobserved and potentially hazardous environment, while continuously receiving and reacting to noisy local observations. This can be cast as a Partially Observable Markov Decision Process (POMDP). Solving a POMDP is PSPACE-Complete [70], so we frame this as a Bayesian Dynamic Motion Planning Problem (BDMP) to impose structure and make the problem tractable. The BDMP problem differs from a POMDP in that uncertainty originates only from the robot's ignorance about the environment. Given the environment, the transition and the reward function and robot's internal state are fully observable [38].

Previous works have exploited this structure via the framework of determinization in the face of uncertainty—repeatedly solving and executing relatively-inexpensive determinized planning problems—with strong theoretical and practical results [38; 94; 56; 126]. Although determinization is computationally efficient, it does not consider what happens when the determinized problem diverges from reality. This can manifest as

optimism (causing collisions) or pessimism (causing roundabout paths, or no path at all).

Our **key insight** is that this deficiency stems from determinization's limited ability to *reason about the distribution of costs over plausible worlds*. Thus, we leverage multi-sample posterior sampling to reap some of the computational benefits of determinization while preserving the planner's ability to reason across multiple plausible environments. Our resulting multi-sample determinization algorithm, Dynamic Replanning via Evaluating and Aggregating Multiple Samples (DREAMS), considers multiple plausible optimal paths and multiple plausible worlds. With this framework, DREAMS enables reasoning not just over the distribution of worlds, but also over additional parameters such as traversal speed. We show that with the correct instantiation, DREAMS outperforms prior determinization strategies on realistic long-range off-road robot navigation tasks.

3.2 *Problem Statement*

We formalize the BDMP problem in our context as follows. Given a start state x_s and goal state x_g in configuration space $\mathcal{X}$ and a set of environments Φ, we seek to minimize the expected total time of traversing from start to goal under the distribution of environments $P(\phi)$. To help solve this problem, we are given a measure of uncertainty over environments modeled as the posterior distribution $P(\phi|\psi_t)$ where ψ_t is the history of observations from time $[0, t]$. This problem extends the BDMP problem to consider the added cost of potential collisions. Collisions can be dangerous—damaging the robot—and require additional time to recover. In practice, $P(\phi|\psi_t)$ would be provided by a perception system.

We focus on planning over roadmaps. Specifically, we are given a graph G with vertices V and edges E. Each edge has a traversal time $w : E \to \mathbb{R}^+$ and a collision status $\phi(e)$ where $\phi(e) = 1$ means e is a collision-free edge in world ϕ. A path $\xi_t = (e_1, e_2, \ldots, e_t)$ is defined as a sequence of edges. Since we are in a dynamic setting, traversing edges adds observations to our history ψ_t.

We consider the following two metrics, Traversal Time and Collision Cost.

- **Traversal Time.** $T(\xi) = \sum_{e \in \xi} w(e)$ is the time it takes the robot to traverse to the goal. Edges in collision are still counted toward traversal time.

- **Collision Cost.** $C(\xi; \rho) = \sum_{e \in \xi} \mathbb{1}(\phi(e) = 0) c(e) \rho(e)$.
  $c(e)$ is the collision cost and $\rho(e)$ adjusts the relative cost of a collision compared to traversal time.

The total cost to reach the goal in one planning episode is:

$$J(\xi; \rho) = T(\xi) + C(\xi; \rho) \tag{3.1}$$

where $\rho(e) = \alpha$ is a constant.

3.3 Related Work

3.3.1 Dynamic Replanning

There has been a fair amount of prior work on dynamic replanning under uncertainty. D* and D* Lite are well-known dynamic replanning search algorithms that have been demonstrated to quickly replan in real-world settings [97; 49; 29]. They both re-use the search tree to rapidly replan when the environment is perceived to change. Neither is designed to reason about uncertainty, and are optimistic about unknown parts of the environment. Re-planning is triggered when the planned path is deemed in collision. In an environment with no sensing uncertainty, this is effective because collisions can be easily determined. In an uncertain environment, D* must either collide with an obstacle to detect a collision or blindly trust its noisy sensors. That being said, D*'s efficient re-use of the search tree could be incorporated into methods that consider environment uncertainty directly.

3.3.2 POMDPs

The BDMP problem can be framed as a POMDP with unknown state, but known transitions and rewards. POMDPs have a plethora of approaches but a notable few rely on a fixed or sampled set of MDPs to make the problem more tractable [79; 89; 94; 100; 15]. Most similar to our approach is DESPOT [94], which samples a set of K scenarios and builds a search tree to alleviate the curse of dimensionality. While promising, solving a POMDP is PSPACE-complete [70], and this work instead focuses on a tractable MDP relaxation with a discrete set of states and known transitions represented as a graph with unknown reward.

3.3.3 Canadian Traveler's Problem

The Canadian Traveler's Problem [71] and specifically its stochastic variants [26; 22; 56; 127] are most similar to the BDMP setting. In both cases, edge collisions are discovered when the agent reaches an incident vertex. This setting follows our observations from real mobile robotic systems: sensing is more accurate near the robot. The stochastic variants additionally use this information to update unobserved edge probabilities. Our setting is similar, except that we discover the true cost of an edge only upon traversal. We also explicitly incorporate noisy sensing. In this sense, our problem is more specific to mobile-robot applications.

3.3.4 Risk-Aware Planning

Risk-aware planning can be seen as another form of planning under uncertainty [17; 8; 101; 30; 65; 13]. Barbosa et al. [8] convert the popular Conditional Value at Risk (CVaR) [110] metric into a cost function for planning and accept new plans only when risk increases along the current trajectory. While these methods have shown great promise, none have investigated planning under the setting of onboard sensing where

uncertainty increases further from the robot. Further, we use a risk-aware cost baseline to demonstrate that treating risk solely as a cost metric fails to produce truly robust plans.

3.3.5 *Determinization*

Determinization, making a deterministic approximation of a stochastic problem, has been effective for planning under uncertainty [126]. In particular, a variety of works [38; 4; 99; 123; 17] have used posterior sampling [105] to make the planning problem tractable and only require sample access to the posterior. Dynamic replanning with posterior sampling (DRPS) [38] samples one problem and solves it optimally, letting sampling naturally balance exploration and exploitation. Sampled A* [17] further utilizes multiple samples per replan and accepts the most likely path, showing promising results. Notably, neither method considers the risk of mismatch between the determinized world and reality. This chapter explores determinization as a method for making planning under uncertainty tractable. Specifically, we look at the effect of mismatch between the determinized world(s) and the real world on planning and how we can combat that.

3.4 *DREAMS*

To summarize various approaches from the literature, we decompose algorithms into a *proposer* and an *acceptor*. The proposer proposes a set of paths $\Xi = \{\xi^0, \xi^1, \ldots, \xi^n\}$. The acceptor considers the proposed paths and accepts one. The robot then follows the accepted path for a step, receives observations, and updates the posterior distribution. It then replans and repeats until the goal is reached. Under this framework, we now describe our approach, Dynamic Replanning via Evaluating and Aggregating Multiple Samples (DREAMS).

3.4.1 *DREAMS Proposer*

The DREAMS proposer is based on posterior sampling, where at each step we sample from the distribution of optimal plans $P(\xi^*|\psi_t) = P(\xi|\phi)P(\phi|\psi_t)$. This is achieved via sampling from the posterior over worlds $P(\phi|\psi_t)$ and then planning the optimal path on each sampled world (Fig. 3.1, *Sample & Plan*). As the robot learns more about the world, this process naturally exploits the knowledge we gain by reducing the spread of the distribution of worlds and plans. Similar to Sampled A*, DREAMS samples multiple plans to approximate the distribution $P(\xi^*|\psi_t)$.

3.4.2 *DREAMS Acceptor*

Unlike prior determinization approaches, we evaluate the cost of each sampled plan against a distribution of sampled worlds (Fig. 3.1, *Evaluate*). The sampled worlds do not need to be the same as the ones used for planning. Empirically, we have found

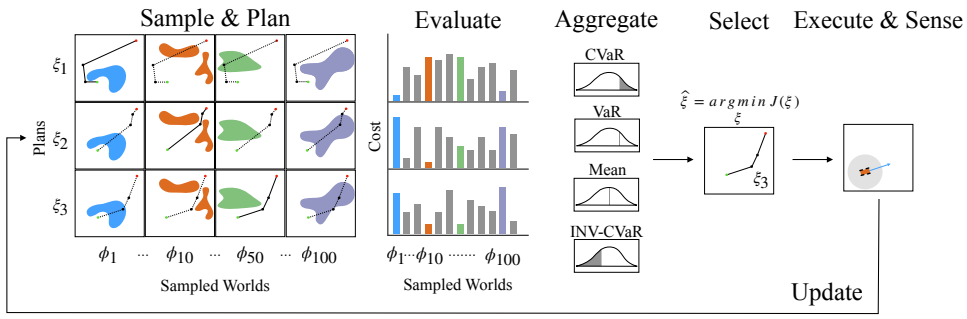


Figure 3.1: **Overview of DREAMS.** *Sample & Plan*: Sample many worlds from the posterior distribution, and find the optimal path on a subsample of worlds (ϕ_1, ϕ_{10}, ϕ_{50} above). *Evaluate*: Evaluate the cost of each plan against the full set of sampled worlds. *Aggregate*: Aggregate the resulting cost distribution with a summary statistic (e.g., mean or CVaR) *Select*: Select the plan with minimal aggregated cost.

that planning is typically the bottleneck rather than evaluation. Therefore, we opt to sample many more worlds for evaluation than for planning.

For each plan, we compute a summary statistic over the resulting distribution of costs (Fig. 3.1, *Aggregate*) and select the plan that minimizes the aggregate cost. Depending on the application, the summary statistic can vary. For example, selecting the minimum cost for a given plan is an optimistic strategy that looks at a plan's cost under the best-case scenario. CVaR summary statistics balance the risk of high cost paths (in this case, collision-prone paths) more carefully. This approach is extremely flexible. In Section 3.5, we additionally use this acceptor to explore different velocity profiles to further reduce expected cost.

3.4.3 *DREAMS Evaluation Function*

We make a small modification to Eq. 3.1 to use as the DREAMS's evaluation function.

$$\widehat{J}(\xi) = T(\xi) + C(\xi;\tau) \tag{3.2}$$

$$\tau(e) = \mathbb{1}(e = e_0)\alpha + \mathbb{1}(e \neq e_0) \tag{3.3}$$

Because DREAMS plans in a receding-horizon fashion, $\widehat{J}(\xi)$ considers the collision factor α only on the immediate edge and assigns a relative cost of 1 to future potential collisions. We found in our setting equally weighting all collisions can create overly conservative behavior due to noisy observations farther from the robot. Reducing future collision cost helps avoid these scenarios. While we focus on traversal time and collision costs relevant to off-road driving, under different settings other cost functions could be used.

3.4.4 *DREAMS Aggregation Function*

We optimistically take the mean of the best 75% of the distribution of costs, calling it the Inverse CVaR. This reduces the effect of unlikely high-cost outliers. Like CVaR, it also considers the width of the distribution: as increased uncertainty causes the cost distribution to spread out and increase the aggregate cost promoting caution.

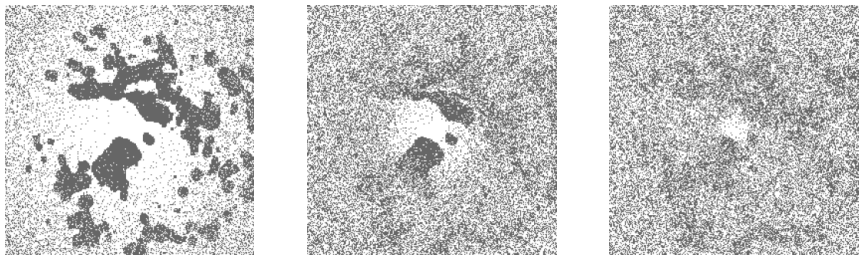


Figure 3.2: **Sensor noise model.** The sensor noise levels used in testing: $\eta_{\text{low}} = 10^{-4}$, $\eta_{\text{med}} = 10^{-3}$, $\eta_{\text{high}} = 10^{-2}$. Our simplified noise model defines the probability of receiving a correct observation for a query point distance d away as $\max(\exp(-\eta d^2), p_{\min})$. The minimum probability threshold $p_{\min}$ is set to 0.6 to provide some signal at the edge of the robot's observation range; note that the minimum possible value of $p_{\min}$ for binary occupancy is 0.5 (pure noise). With these parameters, the robot receives approximately 96%, 72%, and 61% correctly observed pixels per observation.

3.4.5 *Benchmark Overview*

We compare this algorithm with multiple relevant baselines summarized below. Per our framework, each algorithm proposes plan(s), accepts a plan, follows one edge, and replans. Algorithms differ in their choice of proposer and acceptor.

- **DRPS [38].** Proposer: Sample one plan from the posterior. Acceptor: choose only plan.
- **Sampled A* [31].** Proposer: Sample multiple plans from the posterior. Acceptor: choose the most likely plan by computing edge centrality across plans and selecting the path with the highest mean edge centrality. Centrality for an edge is the number of optimal plans that flow through that edge.
- **Direct.** Proposer: Compute the plan that minimizes risk-aware evaluation cost in expectation. Acceptor: choose only plan.

$$\mathbb{E}[\widehat{J}(\xi)] = \sum_{e \in \xi} w(e) + P(\phi(e) = 0) c(e) \tau(e) \tag{3.4}$$

Both DRPS and Sampled A* incorporate posterior sampling and present strong results on similar problems. DRPS highlights the difference between single and multi-sample posterior sampling, while Sampled A* compares the evaluation/aggregation acceptor strategy with an approximate MAP estimate. These baseline algorithms only consider the geometric path at an arbitrary fixed speed. We include results for DREAMS-Fixed to compare most directly with these algorithms, and provide additional results for DREAMS-Adaptive to demonstrate our ability to evaluate paths under different parameters–in this case, speed. Finally, Direct optimizes Equation 3.2 directly to compare with posterior sampling.

3.5 *Experiments*

The following simulation experiments are designed to replicate an off-road autonomous driving scenario where the planner faces limited sensor range and noisy perception. The robot must navigate as efficiently as possible while avoiding dangerous collisions, re-planning at each step. We consider the following hypotheses:

H1. Both DREAMS variants will incur lower total cost compared to DRPS/Sampled A*. More sampled plans will help DREAMS reduce cost variance relative to DRPS.

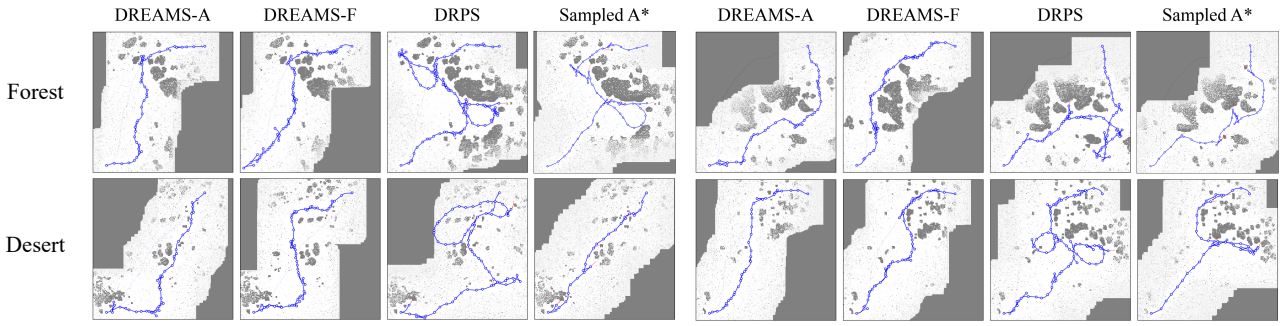


Figure 3.3: **Traversed paths of each algorithm (blue edges) on two example worlds from each of the Forest and Desert datasets**. Each set of four panels shows all algorithms' paths on the same world. Each algorithm receives observations with high noise, and is penalized with a collision factor of $\alpha = 10$. With high noise, DRPS frequently backtracks and changes direction while Sampled A* incurs many collisions (red edges). Both DREAMS variants follow more direct paths without collisions.

Reasoning about a distribution of costs rather than accepting approximate MAP estimate will help DREAMS incur less collision cost than Sampled A*.

H2. DREAMS-Adaptive will incur lower collision/total cost compared to DREAMS-Fixed. Reasoning about the distribution of speeds allows the vehicle to slow down when the likelihood of a collision increases and speed up when the path is likely free.

H3. Increasing the number of sampled plans will reduce the total cost (with diminishing returns). More plans will provide more options to evaluate, until all likely options are enumerated.

H4. Increasing the number of sampled worlds considered during evaluation will reduce the total cost (with diminishing returns). Sampling more worlds will yield a more accurate estimate of expected cost.

H5. DREAMS will incur lower total cost compared to Direct. To plan directly with the cost function, Direct replaces the indicator in Equation 3.2 with a probability. This can result in extremely unlikely plans under the posterior $P(\xi^*|\psi_t)$, which DREAMS probabilistically avoids by construction.

3.5.1 Experimental Setup

3.5.2 Real World Occupancy Grids and Speeds

We evaluate performance with two real-world datasets of long-range planning problems through open desert environments ($N = 83$) and more challenging crowded forest environments ($N = 51$). These datasets were collected by running a perception stack on a full-scale off-road vehicle as part of the RACER program [20].

Each world spans 100×100 meters at a resolution of 0.4 m/px. The robot moves on a graph covering the space at speeds ranging from 1–10 m/s. As the robot traverses, it receives observations at 1 Hz. Therefore, speed affects both traversal time and the number of observations received. The robot may reverse at 1 m/s, but we first require planners to find a forward-only solution; if none exists, reversing is then permitted. We evaluate on ten random seeds for each world ϕ, noise level η, and collision α.

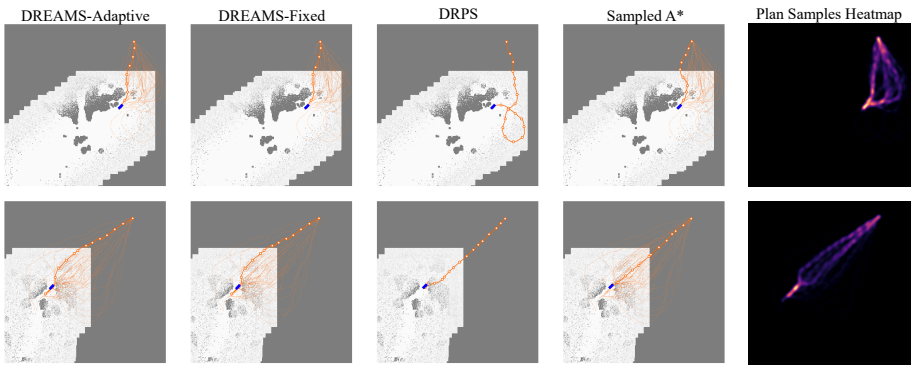


Figure 3.4: **Qualitative comparison of each approach, given the exact same sampled worlds and paths.** Robot (blue), proposed paths (light orange), accepted path (bright orange). *Top:* All plans except DRPS find a path through the gap. DRPS samples a world in which the gap is blocked, producing a longer route. *Bottom:* All plans except Sampled A* reverse from a likely obstacle in front of the robot. Sampled A* does not explicitly consider the cost of collision and accepts a path going through the obstacle. *Right:* Looking at the heatmap, areas where more plans overlap are hotter. As there is little overlap besides the start position, Sampled A* has less signal to choose the most likely plan; its decision is almost a uniform random sample.

3.5.3 *Limited Range Noisy Observations*

Many autonomous systems process raw sensor input into semantic classification of the environment using a deep neural network [86; 82]. Noisy sensors and limited training introduce uncertainty into the resulting semantic segmentations. Generally, the predictions become noisier farther away from the robot where there is less (and noisier) sensor information. Predicted semantics eventually become too noisy and are thus limited to a reliable range. To emulate this effect, we introduce a limited range observation module that simulates the classification of obstacles. The robot can only observe a patch of 50×50 meters centered around itself. Fig. 3.2 describes the sensor model within this limited range observation.

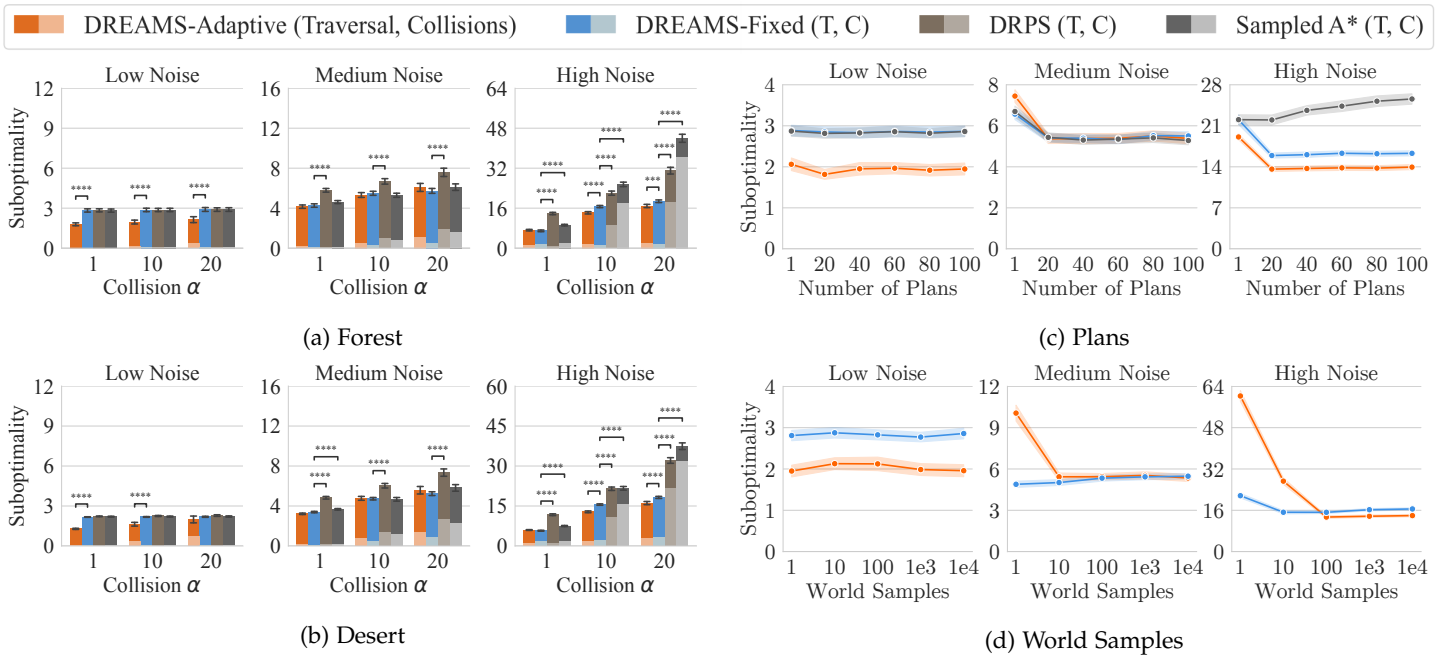
Outside of the limited range observation, we optimistically assume that the space is free to allow posterior sampling to discover many plausible paths. This is similar to how real-world systems work, where unknown space is a fixed cost. Chapter 4 proposes an alternative method to this fixed cost. The posterior is updated using Bayes' Rule.

3.5.4 *Posterior Sampling*

To sample from the posterior distribution over optimal plans $P(\xi^*|\psi_t)$, we first sample from the posterior distribution over worlds $P(\phi|\psi_t)$ and plan on each world. We sample a world by sampling over the posterior distribution of roadmap edges. For each edge, we assign its collision probability as the maximum posterior probability over all pixels covered by the robot's swept volume (3.5×1.5 meters) along the edge.

3.5.5 *Planning and Execution Parameters*

For DREAMS and Sampled A*, we choose to plan with 100 posterior samples using A*. DREAMS evaluates each plan according to (Equation 3.2) against 10^4 sampled worlds. DREAMS-Adaptive additionally considers multiple speed profiles. For each sampled plan, it creates five timed trajectories each with a separate speed profile. All profiles use 5 m/s inside the observed area and 10 m/s outside; they differ only in their first-edge traversal speed, which takes values in {1, 3, 5, 7, 10} m/s. DREAMS-Adaptive



(a) Forest

(b) Desert

(c) Plans

(d) World Samples

Figure 3.5: **Course and ablation metrics.** *Left:* Suboptimality plots for (a) Forest and (b) Desert datasets. We perform a Welch's t-test for difference of means, with a Bonferroni correction of 90 for all pairwise comparisons involving DREAMS. * : $p < 0.01$, ** : $p < 0.001$, *** : $p < 0.0001$, **** : $p < 0.00001$. *Right:* Ablation study for varying (c) number of sampled plans and (d) number of world samples in evaluation. (Error bars in both figures denote 95% confidence intervals.)

executes the chosen speed for the chosen plan for one time step before re-planning. Like DRPS and Sampled A*, DREAMS-Fixed uses a single profile: 5 m/s inside the observed area and 10 m/s outside.

The collision cost is proportional to the robot's pre-collision speed (Equation 3.1), as higher speed collisions are more dangerous. We vary $\alpha \in \{1, 10, 20\}$ to assess the impact of varying collision cost weight.

3.5.6 *Metrics*

We compare each algorithm's cost to the cost incurred by an oracle ξ^{opt}, which has full information about the world and traverses at 10 m/s without collisions.

$$\text{Suboptimality} = J(\xi)/J(\xi^{opt}) = J(\xi)/T(\xi^{opt}) \tag{3.5}$$

3.5.7 *Results*

Fig. 3.5a and 3.5b show suboptimality results for the Forest and Desert datasets. An ablation study with the more challenging Forest dataset is shown in Fig. 3.5c and 3.5d.

Qualitative results of traversed paths on the final posterior are shown in Fig. 3.3. Fig. 3.4 compares each algorithm under the exact same scenarios. Table 3.1 gives planning time results. **H1. Both DREAMS variants will incur lower total cost compared to DRPS/Sampled A*.** In both datasets (Fig. 3.5a and 3.5b), DREAMS-Fixed is competitive with DRPS and Sampled A* in Low noise. It is either competitive or outperforms them in Medium noise. In High noise, it dominates for all α values tested. H1 is supported. It is not surprising that all algorithms achieve similar results in Low noise, as the sampled plans will likely be near-optimal (i.e., most sampled worlds are close

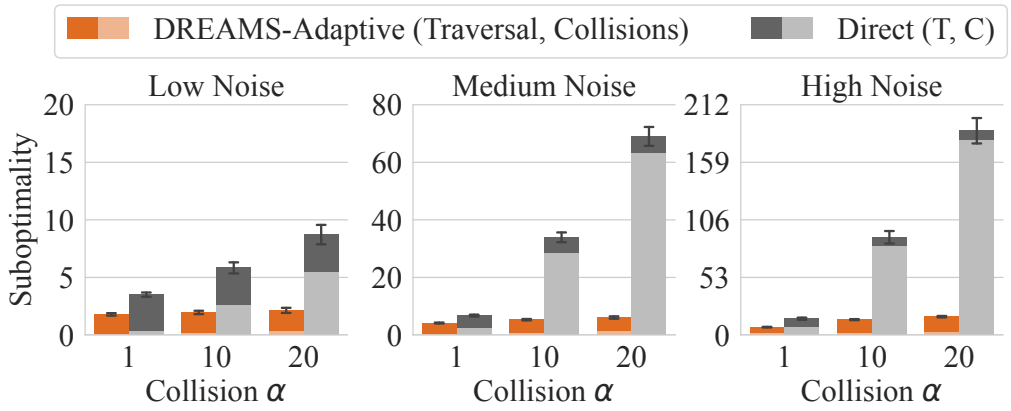


Figure 3.6: **Comparison of DREAMS to Direct on the more challenging Forest dataset.** Direct incurs many more collisions because it does not reason about the likelihood of the plans directly. Note: The suboptimality axis is much higher than Fig. 3.5a.

to the true world). In Medium noise, we observe the benefit of multiple samples as both DREAMS variants and Sampled A* perform better than DRPS. But in High noise, Sampled A* incurs a high collision cost as it does not explicitly reason about collisions; all plans seem equally likely because the distribution of plans is spread out, showing less benefit to multiple samples without explicit collision reasoning. Fig. 3.4, bottom shows an example scenario of this behavior.

H2. DREAMS-Adaptive will incur lower collision/total cost compared to DREAMS-Fixed. In Fig. 3.5a and 3.5b, we see a mixed result between DREAMS variants: DREAMS-Adaptive has statistically-significant lower cost in some cases but not all. With Low noise, DREAMS-Adaptive can move faster reducing its traversal time without incurring too much collision cost. In Medium noise, DREAMS-Adaptive and DREAMS-Fixed achieve similar performance; this suggests that these higher speeds do not properly balance speed and safety. In High noise, DREAMS-Adaptive seems to trade-off collisions and traversal better. Because the results are mixed, H2 is not strongly supported.

H3. Increasing the number of sampled plans will reduce the total cost (with diminishing returns). Fig. 3.5c shows that more sampled plans reduce the total cost across multiple noise values for DREAMS. The leveling off at 20 plans shows that the sampled set is fairly representative of our posterior distribution. H3 is supported. Surprisingly, Sampled A* performed worse with increased samples at High noise. We attribute this to High noise causing a spread out distribution of plans where no plan has a large mean centrality, resulting in near random choice (See Fig. 3.4 for an example of this). Increasing the number of plans introduces more options for Sampled A* to choose from; if these are likely to be in collision, this will generally increase the cost incurred.

H4. Increasing the number of sampled worlds considered during evaluation will reduce the total cost (with diminishing returns). Fig. 3.5d shows more world samples in evaluation improve performance for DREAMS-Adaptive at Medium and High noise. For DREAMS-Fixed, we only see change in High noise. In this setting, it suggests a few samples captures the true cost well in Low and Medium noise, but more samples are needed in High noise. H4 is supported for High noise.

H5. DREAMS will incur lower total cost compared to Direct. Fig. 3.6 shows the suboptimality comparison between DREAMS-Adaptive and Direct. Direct incurs a very high collision cost and total suboptimality because it finds paths with a high likelihood of collisions. We attributed this to the probability of collision being a multiplicative factor instead of an actual probability, meaning Direct may choose plans that are highly unlikely but have a low total cost. H5 is supported.

Algorithm	Proposer Time (s)	Acceptor Time (s)	Total Time (s)
DREAMS-Adaptive	1.55 ± 0.09	1.59 ± 0.01	3.14 ± 0.09
DREAMS-Fixed	1.54 ± 0.09	0.33 ± 0.00	1.88 ± 0.09
DRPS	0.02 ± 0.00	0.01 ± 0.00	0.03 ± 0.00
Sampled A*	1.47 ± 0.08	1.12 ± 0.02	2.58 ± 0.08

Table 3.1: **Planning time evaluations for each algorithm, on the same set of 300 evaluation runs.** *Proposer:* DREAMS and Sampled A* plan 100 paths per iteration, while DRPS plans a single path. Because there is no dependency between planning each independent posterior sample, this can be trivial accelerated by parallelization (results sequential). *Acceptor:* Since DRPS only proposes one path, the acceptor time is negligible. DREAMS-A evaluates five speeds taking more time while DREAMS-F evaluates just one. Aggregating edge centrality across plans is also relatively expensive for Sampled A*. Results were obtained on a i9 CPU.

3.6 *Limitations*

While DREAMS is a promising step to enabling efficient uncertainty reasoning, there remain a few limitations. First, DREAMS relies on the ability to sample potential costmaps. A potential costmap should be plausible given the distribution of worlds. This requires a perception system that can provide more than per-pixel cost uncertainty, because sampling pixel-wise treats the pixels as independent and can create unrealistic costmaps with snow-like noise. For DREAMS to work on a real system, we think a generative model like a diffusion model that creates whole costmaps given sensor data or inpaints a partial map would be the most effective way to sample unique costmaps.

Second, determinization can cause plan variance over time, where the planned path changes significantly from the previous timestep. This occurs because worlds sampled at each time step can differ substantially from those in the previous step. While sampling more worlds can mitigate the problem, plan variance remains an issue. Future work could look into maintaining multiple likely worlds that are temporally consistent, or considering switching costs in plan evaluation.

3.7 *Research Questions*

RQ1 What is the effect of mismatch between the determinized world and the true world?

While determinization can make planning under uncertainty a tractable problem, it comes with the risk of mismatch between the determinized world(s) and the true world. For example, the determinized world could lack an obstacle that exists in the real world and cause a collision.

We find determinization can have three effects of mismatch: over-optimism, over-pessimism, and plan variance. Over-optimism occurs when the sampled world has falsely identified part of the environment as free space and the resulting plan goes

through that space. Over-pessimism is the opposite case where false obstacles exist in the determinized world causing the planner to find unnecessarily roundabout plans. Further, false obstacles and freespace can both exist in the same determinized world. Fig. 3.7 *Top Left* shows examples of over-optimism and over-pessimism in a simulated environment. Mismatch increases with more sensing noise, and Fig. 3.7 *Top Right* shows the effect of that mismatch in both increased collisions and traversal time. Finally, plan variance can occur in some forms of determinization where determinized worlds across multiple timesteps have little consistency, which causes the plans to fluctuate. While this is not exactly an effect of mismatch, it remains a challenge for determinization strategies.

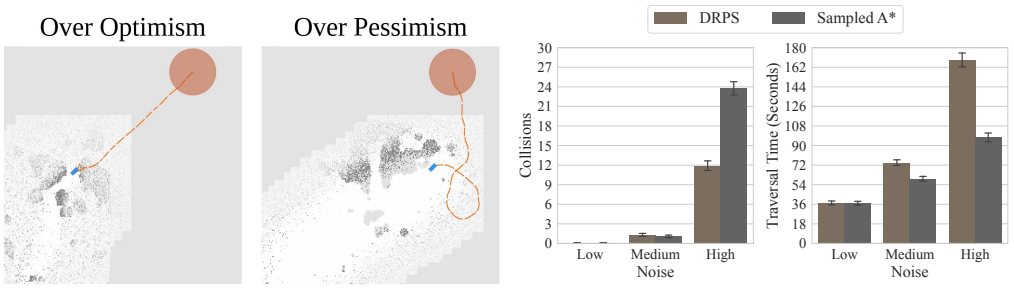


Figure 3.7: **The effect of determinism mismatch.** *Top Left:* Examples of over optimism and pessimism. Over optimism leads to collisions. Over pessimism leads to unnecessarily roundabout paths increasing traversal time. *Top Right:* Collisions and traversal time plots as sensor noise increases for two determinization approaches.

RQ2 How can we combat the effects of determinization mismatch?

Although determinization is computationally efficient, it does not account for when the determinized problem it solves diverges from reality. Mismatch between the determinized and true worlds can cause collisions, roundabout paths, and path variance.

To address this problem, our key insight is that this deficiency stems from determinization's limited ability to reason about the distribution of costs over plausible worlds. Thus, DREAMS leverages multi-sample posterior sampling to reap some of the computational benefits of determinization while preserving the planner's ability to reason across multiple plausible environments.

We find that DREAMS, by evaluating multiple candidate plans and their cost distributions, can mitigate these mismatches. By considering the cost in different worlds than the ones planned on, we can choose the path that performs well across worlds. From simulated experiments on real-world costmaps, we show DREAMS outperforms other methods, particularly at high noise by balancing optimism and pessimism about the true world.

4
Leveraging long-range Sparse Sensor Information

This chapter addresses research questions RQ3, RQ4, leveraging long-range sensor information for planning. Specifically, this research uses sensor data that lies outside the robot's local costmap, enabling reasoning about distant terrain instead of treating it as "unknown."

4.1 Introduction

Autonomous off-road mobile robots require long-range waypoint navigation, often in environments where prior information (e.g. satellite imagery) is inaccurate or unavailable. Our goal is efficient navigation in these large-scale scenarios where our domain is significantly (10X or more) larger than the robot's sensor footprint. The central research question is,

How can we enable robots to make less myopic decisions facilitating long-range navigation with only incomplete knowledge of the environment?

We assume that the robot has GPS localization together with target waypoints. Notably, the robot is provided with no other information about the environment and must find its own path to the goal with minimal human intervention and as fast as possible. Mobile robots with no prior information traditionally create a metric cost map based on onboard sensory information (e.g. cameras, lidar, and odometry) [60; 73]. This suffices for short-horizon targets, but with long-range goals, creating large cost maps is intractable due to limited sensor range, compute, and memory. In particular, additional depth information is required to project features into the map, which is often sparse and noisy, resulting in a limited map horizon and the area outside this horizon being a fog of unknown space. A go-to approach to deal with unknown space is to heuristically assign it a fixed cost, effectively planning straight to the goal once outside the map [73]. However, this leads to highly inefficient and potentially unsafe paths in many scenarios. A **key observation** is that field operators can determine the robot's long-range strategy by simply analyzing its image feed, without the requirement of a complete terrain map. For example, it is possible to spot the opening in a wall of trees from images without mapping every tree. We refer to the boundaries between

known/unknown regions as *frontiers*. Frontiers which visually appear open i.e., possible to navigate to and continue beyond are referred to as *affordable* frontiers. With this, we offer the following **key insight**: A robot can reason further by learning to identify distant *affordable* frontiers as intermediate goals.

We propose improvements to current heuristic-based approaches by learning affordable frontiers in the image space. Specifically, we leverage the SAM2 foundation model [77], pre-trained on a large corpus of image data, to generate meaningful camera image embeddings. These embeddings capture notions of scene segmentation, invariant to lighting, camera angle, and texture. We train a small, `LRN`-specific decoder to predict long-range affordance heatmaps in a *goal-agnostic* manner. We then project the heatmaps into heat scores for different headings the robot could follow and re-weight each heading based on the goal context. The top scoring direction is passed to the local system as a heading to follow to reach the goal. We refer to our method as **Long Range Navigator (`LRN`)**, depicted in Fig. 4.1.

4.2 Problem Statement

We assume the robot perceives its environment from its local sensors (e.g. camera, lidar), and is tasked with navigating to a distant goal $g \in \mathcal{G}$ in a static, unknown environment. The robot is equipped with access to a local policy $\pi_\lambda : \mathcal{S} \to \mathcal{A}$, that maps the state s to primitive actions $a \in \mathcal{A}$ and plans under a cost function $C : (s, a, s') \to \mathbb{R}$. Planning is limited to some horizon H due to sensor or compute limitations. We assume the policy receives an observation o, plans, executes an action, and replans at some frequency in a model predictive control fashion. As it executes actions, the robot creates a path ξ^π. The **objective** is to minimize the expected cost of navigating to the goal in an unknown environment ϕ.

$$J(\xi^\pi) := \arg\min_{\xi^\pi} \mathbb{E}_{P(\phi)} \left[C(\xi^\pi) \right]$$

4.3 Related Work

4.3.1 Near-to-far Learning

Near-to-far learning is a well-explored technique for extending the sensing range beyond dense sensing required for traditional Cartesian maps [27; 62; 63; 7; 108; 129]. It relies on two key observations: cameras provide long-range sensor information, and sensing near the robot is reasonably accurate. As such, these approaches train an image classifier to distinguish traversable and non-traversable terrain. Training is self-supervised by using accurate near-range sensing to provide labels for previous timesteps when the observed region was far away from the robot. Once classified, a planner is run directly in the image space [116; 66; 69] or image regions combined with sparse stereo data are down-projected into a polar perspective map [7] that enables use of a geometric planner. Image space planning enables longer ranges beyond stereo limits but suffers from common failure cases such as occlusions and small localization

errors. Projection does not suffer from these issues (because of depth information) but is limited to the range of depth sensors, and far-away map cells become coarser, preventing detection of small openings like paths through trees. More recently, FITAM [27] forgoes the need for depth and directly predicts the likelihood of various cost classes for cells by learning a mapping from vertical image slices to cell classes. This shows depth is not necessarily needed for coarse (25 m) mapping. However, like other near to far methods, this approach can still miss narrow openings due to map coarseness.

4.3.2 *Subgoal Planning*

These works [96; 33; 124] focus on planning to subgoals or frontiers. Most similar to our work is Stein et al. [96] which learns whether a subgoal will reach the goal and the cost to go. The model is trained in simulation to map raw lidar sensor measurements to cost to go, probability of reaching the goal, and exploration cost. The full ground truth map is used in conjunction with a motion planner to get ground truth information. This work was the first of its kind to show that one can learn a mapping from raw sensor data to viable frontiers. In this work, we aim to take the same ideas but apply them to different sensor data (cameras) and in outdoor environments. One work that uses camera data [76] learns where the robot can and can't go by backprojecting where the robot went into the image space in video game worlds. This shows promise that cameras can be used effectively for subgoal selection. We would like to leverage a similar approach but in real-world scenarios without perfect depth data to enable backprojection.

4.3.3 *End-to-End Visual Navigation*

Visual navigation is the task of navigating a robot primarily from camera inputs. These approaches generally predict a target point ahead of the robot to navigate to [59; 87; 95] and may use a series of images to build a topological map [81]. These approaches are notably myopic [59] or rely on pre-collected topological maps [87; 95; 81]. Finally, all of these approaches are applied to smaller scale tasks at a maximum of 1 km from the target whereas we are focused on much longer range tasks in the scale of tens of kilometers. We show the limitations of these methods via comparing against NoMaD [95].

4.3.4 *Traversability Prediction*

These works [32; 82; 46; 119] predict traversability in image space. While this is a similar task to ours, traversability alone is not sufficient to find affordable frontiers. We show this via comparing with the Traversability + Depth Anything V2 baseline. Further, many of these approaches are trained in a self-supervised fashion by projecting robot ego-trajectories into image space, similar to our training approach. We found empirically projecting 2D trajectories can be sufficient for local traversability, but is too noisy (small localization errors cause large shifts) for finding distant frontiers – hence we instead opt for image tracking to get ego-trajectories.

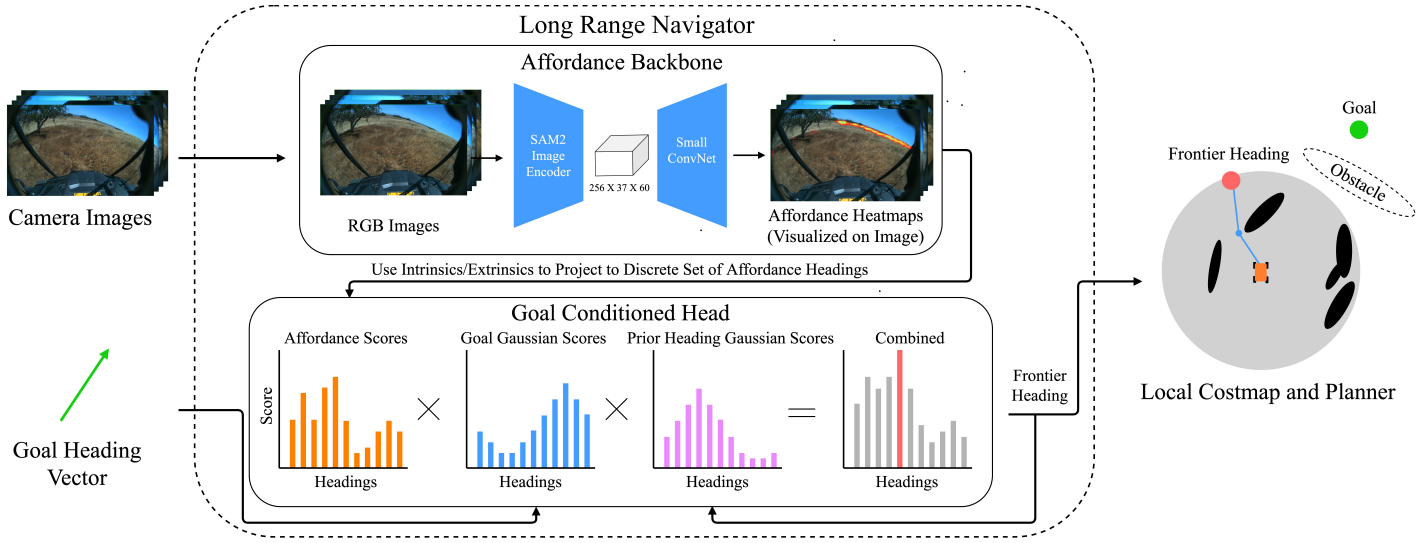


Figure 4.1: **Overview of LRN.** LRN is fed with egocentric camera images and a goal heading vector. It is composed of the following components, 1) **Affordance Backbone**: computes *affordable* frontiers in the image space as heatmaps agnostic of the goal. These affordance hotspots are then projected into a discrete set of affordable headings for the robot to follow, 2) **Goal Conditioned Head**, wherein the affordance scores are multiplied with a discrete gaussian score around the goal and a separate gaussian around the previous prediction (to maintain consistency). The maximum combined score heading (red) is selected. The local system can then use that frontier as a goal for local planning instead of the true goal. This process repeats as new sensor information comes in.

4.3.5 *Cost Inpainting*

These methods [86; 27; 60] attempt to inpaint cost in unknown space. This works particularly well because the model can leverage local context in making its predictions. Notably [27] uses a diffusion model to predict a larger map given the local costmap but does not use sensor information besides the costmap for prediction.

4.3.6 *Learning From Videos*

Prior works [52] demonstrate learning navigation subroutines from ego-centric videos similar to our training scheme. However, these are more short horizon subroutines like turn-left or go-through-door and do not reason about distant unknown space.

4.4 *Long Range Navigator (LRN)*

Core to our approach is the idea that, although the low-level controller policy π_λ is limited to a local horizon, useful sensor data beyond that horizon can still aid navigation. Consider a set of frontier states $f \in \mathcal{F}$ at the periphery of the horizon H, which borders known/unknown space as shown in Fig. 4.2. Under the optimal substructure property [19], if we know the optimal frontier node f^*, which lies on the optimal path from start to goal and π_λ is optimal up to H then π_λ planning to f^* is acting globally optimally. In practice, this is very hard, because π_λ will usually be sub-optimal and f^* depends on unknown information not detected in any of the robot's sensors (e.g., backside of a hill). That being said, there may exist information within the robot's sensor range that can help estimate affordable frontiers, which seem possible to navigate to and continue beyond.

To estimate affordable frontiers from state s, we first define the value of a frontier f given a goal g as $V(s, g, f)$. This can be decomposed into two parts, namely $A(s, f)$

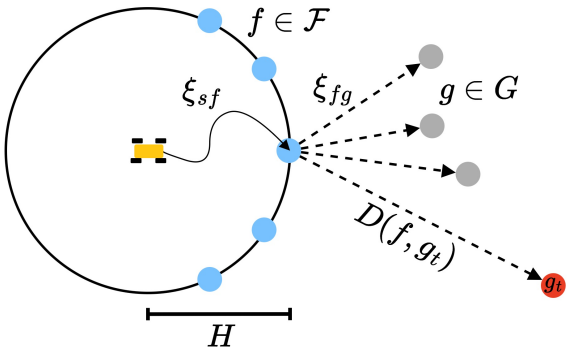


Figure 4.2: **LRN's formulation of the long-range navigation problem.** LRN learns the value estimate $V(s,g,f)$ using affordability score $A(s,f)$ for each frontier and the cost to goal estimate $D(f,g_t)$.

and $D(f,g_t)$, as shown in Fig. 4.2. For clarity, a specific navigation goal at time t is denoted g_t. Formally,

$$V(s,g,f) = A(s,f)\, D(f,g_t) \tag{4.1}$$

$$A(s,f) = P(\exists \xi_{sf})\, P(\exists \xi_{fg}, \exists g \mid d(s,g) > H), \tag{4.2}$$

where $A(s,f)$ measures the *affordability score*. This score is computed by multiplying the probability there exists a path from s to f and the probability there exists some path from f to some distant goal g beyond the local horizon H. $D(f,g_t)$ measures the cost estimate of navigation from the frontier f conditioned on the goal. Given, $V(s,g,f)$ and π_λ, we can define the policy we seek π.

$$f_\pi = \arg\max_{f \in F} V(s,g,f) \tag{4.3}$$

$$\pi(s,g) = \pi_\lambda(s, f_\pi) \tag{4.4}$$

LRN is a bi-level system, with two components: one to estimate $A(s) = [A(s,f)]_{f \in F}$ and the other to estimate $D(f,g_t)$. We implement the first component $A(s)$ to estimate *all affordance scores* via a learned mapping from camera sensors to affordances (selective attention) in the image space. The second component scores frontier states given a goal context. The overall algorithm is depicted in Alg. 1. In practice, we discretize the space around the robot into angular bins, which constitutes the space of frontiers $\mathcal{F}$.

Evaluating world space frontiers using image data can be challenging due to potential projection errors or occlusions when trying to project frontiers into the image space. Besides, frontiers may not clearly associate with important features in the image. For example, there may be a distant opening in the trees that the local costmap has not yet reached. The frontier may be in an open area far from the treeline, so evaluating its affordability will require relating it to information far from the frontier in image space. Further, we would like to leverage pre-trained foundation models to maximize performance, and many like SAM2 [77] and DINO [67] operate in image space. For these reasons, we propose instead to learn an intermediate representation of affordable image frontiers and project them to local frontiers.

Image frontiers are converted to (world) frontiers by projecting them to a ray using camera intrinsics (we assume no reliable depth is available) and selecting the point at a distance H along that ray. The projected image frontier is not where the image point truly is in 2D space. So to make this projection reasonable for navigation, the affordable image frontier must have a clear line of sight path from the projected point at distance H to the true 2D point associated with the image frontier. This property is impossible to enforce perfectly without a prior map. Instead, we approximate it by learning a mapping from images to affordance heatmaps via automatic data labeling and human annotations.

Algorithm 1 `LRN`: Long Range Navigator

Require: k angular bins, start s_{start}, goal g, goal stdev σ_g, prev stdev σ_p, EMA α

Phase I- Supervised Pre-Train Affordance Backbone
Input Dataset $\mathcal{D}_v$ of ego-centric videos
Track $\mathcal{D}_v$ into $\mathcal{D}_\xi$ of trajectories
Convert $\mathcal{D}_\xi$ into $\mathcal{D}$ of (image, heatmap) pairs
Train $A(s)$ on $\mathcal{D}$ via supervised learning
Phase II- Online Control with Dynamic Planning
$s \leftarrow s_{start}$
$\mathbf{p} \leftarrow [1]_{i=1}^{k}$
while $s \neq s_{goal}$ **do**
    $\mathbf{b}_{filtered} \leftarrow$ `Affordance_Backbone`(s)
    $\mathbf{g} \leftarrow \left[\mathcal{N}(x_i; g, \sigma_g)\right]_{i=1}^{k}$
    $\mathbf{v} \leftarrow \mathbf{b}_{filtered} * \mathbf{g} * \mathbf{p}$
    $f_\pi \leftarrow \arg\max(\mathbf{v})$
    $\hat{a} \sim \pi_\lambda(s, f_\pi)$
    $s \leftarrow \text{Execute}(s, a)$
    $\mathbf{p} \leftarrow \left[\mathcal{N}(x_i; f_\pi, \sigma_p)\right]_{i=1}^{k}$

Algorithm 2 `Affordance_Backbone` $A(s)$

Require: EMA α

heatmap $\leftarrow$ `PredictHeatmap`(s)
$\mathbf{b} \leftarrow$ `Project`(heatmap)
$\mathbf{b}_{norm} \leftarrow \mathbf{b} / \sum_{i=1}^{k} b_i$
$\mathbf{b}_{filtered} = \alpha * \mathbf{b}_{norm} + (1 - \alpha) * \mathbf{b}_{filtered}$
`Return` $\mathbf{b}_{filtered}$

4.4.1 *Learning Affordances*

To train the affordance backbone we frame it as a supervised learning problem. We keep the SAM2 image encoder frozen and only train a small convnet that maps the SAM embeddings to a grayscale heatmap image at full resolution. The model is trained

via a an mean squared error (MSE) loss with L2 regularization. MSE is computed between the predicted scores and the ground truth heatmaps with values ranging from 0 (not affordable) to 1 (affordable frontier). Note, we used SAM2 [77] for Racer Heavy tests and the lighter-weight MobileSAM [128] for Spot tests due to compute limitations.

To get the ground truth affordance heatmaps we explored both hand labeling and auto labeling. For hand labeling, we asked a human to select points in a given front facing image that satisfy three qualities:

1. Points are at the edge of visible space.

2. There is a reasonably high likelihood of the given robot being able to reach said point. For example, the point isn't behind an impassible obstacle.

3. There is a reasonable likelihood of the robot being able to continue beyond that point. For example, the point is on the crest of a hill not in front of a wall.

Humans selected points via a simple GUI (Fig. 4.3) that provided additional sensor data, but only required labeling the front image. Example human labels compared to learned heatmaps can be found in Section 4.6.4.

Figure 4.3: **Labeling tool for affordance maps.** The red lines in the center image are labeled affordable frontiers. Side images are for additional context but do not get labelled.

4.4.2 *Learning Affordances from Unlabeled Videos*

While we show results from learning image affordances from hand-labeled data (Fig. 4.10), this approach proved to be tedious and scales poorly with more data. This raises the question - *can we learn such affordances from unlabeled videos*? Concretely, we utilize unlabeled ego-centric videos to generate affordable image frontiers. The **key insight** here is that to generate labels analogous to human-annotation, the end of a trajectory should be representative of a good frontier from the starting state of this trajectory. To get such trajectories in image space we leverage the video based tracking scheme discussed in Chapter 5.

Once points are tracked, heatmaps are generated by taking the affordable frontier image points and marking them as 1 (affordable hotspot) and the rest of the trajectory as 0 (not affordable frontier). All other pixels remain unlabeled, and predictions there do not contribute to the loss. We found further marking the vertical column around the affordable hotspots as 0 reduced false positives, particularly in the sky.

4.4.3 *Goal Conditioning*

Given the affordance heatmaps from each camera, we project the scores into a discrete set of angular bins via the camera's intrinsics. For each bin LRN takes the sum of scores falling in that bin for the given camera. Bins with scores less than a threshold h_{thresh} are set to 0. The max is taken for overlapping bins between cameras, then the scores are normalized. Finally, a exponential moving average (EMA) filter is applied with a weight α to filter scores over time reducing fluctuations. If a vector of k discretized bin scores is denoted as $\mathbf{b}$, then

$$\mathbf{b} = [b_1, b_2, \ldots, b_k] \qquad \mathbf{b}_{\text{norm}} = \frac{\mathbf{b}}{\sum_{i=1}^{k} b_i}$$

$$\mathbf{b}_{\text{filtered}} = \alpha \cdot \mathbf{b}_{\text{norm}} + (1 - \alpha) \cdot \mathbf{b}_{\text{filtered}}$$

The goal conditioned cost function (Fig. 4.1) takes in $\mathbf{b}_{filtered}$ the goal angle μ_g and the previous selected heading μ_p. The goal heading and the previously timestep's selected heading each define a Gaussian score centered on μ_g and μ_p previously. Similar to how the EMA filter reduces fluctuations in the individual scores, the previously selected heading is used to reduce fluctuations in the final selected heading. We apply these functions to the discrete bins to obtain a goal cost vector and a consistency cost vector for k angular bins.

$$\mathbf{g} = \left[\mathcal{N}(x_i; g, \sigma_g)\right]_{i=1}^{k} \quad \mathbf{p} = \left[\mathcal{N}(x_i; f_\pi, \sigma_p)\right]_{i=1}^{k}$$

Where, σ_g and σ_p are both fixed parameters. Finally all the vectors gets multiplied together to obtain final scores. The maximum scoring angle is then selected.

$$f_\pi = \arg\max(\mathbf{b}_{\text{filtered}} \cdot \mathbf{g} \cdot \mathbf{p})$$

4.5 *Experimental Design*

In this section, we instantiate LRN in two practical setups—Spot and a Racer Heavy platform—both operating outdoors. We designed our experiments to answer the following research questions:

[Q1.] Can the intermediate affordance representation proposed in LRN exhibit more efficient navigation capability as compared to other approaches? (See Sec. 4.6.1)

[Q2.] Considering the connection between the quality of the affordance model against system performance, do better affordances lead to more efficient paths? (See Sec. 4.6.2)

[Q3.] How does auto-labeling versus human labeling affect affordance quality? (See Sec. 4.6.3)

4.5.1 *Local Policies*

For both platforms' local policies, we follow a traditional perception, planning, and control pipeline where perception creates a metric costmap, planning finds a path through it, and control roughly tracks that path re-planning with new information.

Figure 4.4: **Example Heatmaps computed by LRN.** The two images on the left show examples from the Racer Heavy. Blue is lower confidence and red is high confidence. LRN finds affordable spots between trees and on hill crests. On the Spot dataset on the right LRN correctly puts high score on the two forks in the road and to the side of bushes. For more qualitative results see Section 4.6.4.

Spot We leverage Elevation Mapping CuPy [61], a fast elevation mapping software. The local elevation map is a square with 16 m width/height and created from a combination of depth cameras and an Ouster OS1 lidar . The elevation map is converted to a costmap for planning by mapping slopes to cost via simple rules. We then use an ARA^* planner [55] over a lattice to plan a path through that map. A carrot point 3 m ahead along the planned path is used to compute a body frame velocity, which is passed to Spot's internal navigation system. The internal navigation system handles locomotion and performs some obstacle avoidance.

The entire stack and LRN are run on a Jetson Orin AGX in real-time. To achieve real time performance LRN uses a MobileSAM image encoder [128] and achieving a 4 Hz inference with autonomy also running. At runtime LRN performs inference on both raw front and rear fisheye lens from a Insta360 camera. Once a heading is computed, it is sent to the planner as a goal just outside of the costmap boundary.

Racer Heavy We deployed on a Racer Heavy platform. It is a 12 ton tracked vehicle equipped with three front-facing cameras and one rear camera amongst other lidar and odometry sensors. The local stack in this demonstration was a heavily optimized perception planning and control stack with a circular costmap of radius 50 m. The search-based planner will plan to the goal but stop once it reaches a frontier node at the edge of the costmap within a short tolerance. This allows the planner some flexibility in case an obstacle is blocking the exact goal. LRN was run on all four time-synced camera images. We opted for a larger SAM2 [77] image encoder which ran at 4hz with the autonomy stack running.

4.5.2 *LRN Training*

Spot We collected data in two semi-urban environments using an Insta360 camera. In total, we recorded 54 minutes of video. Because the points of interest lie on the ground, tracked points can drift over long durations. In order to have clean data, we chopped the videos into 2 minute segments. The videos were then processed by our automatic labeling pipeline to produce a dataset of 92,711 heatmap labels.

Racer Heavy The Racer Heavy experiment used an early version of LRN trained on human labeled data. The dataset was 1,901 human labeled heatmap images from a California oak savanna. Notably, the labeled images were only front-facing camera im-

ages from a different Racer platform than our deployment platform. We asked humans to label all affordable frontiers in each image. Human selected regions that were positive labels and the rest of the image is assumed negative. We additionally added some Gaussian blur around positive labels, as we found the smooth transitions helped with training. To improve robustness to visual variations, we further augmented the dataset by applying random color jitter, sharpness adjustments, rotations, and blur, increasing the dataset size to 11,406 images.

4.5.3 *Goal Conditioned Head*

Spot For the goal-conditioned level we use an angle discretization of 5 degree width bins, $h_{thresh} = 0.7$, $\alpha = 0.1$, $\sigma_g = 90$, and $\sigma_p = 110$. Additionally, to avoid walking past the goal we implement two additions. First, when the robot is within 30m of the goal it linearly reduces the σ_g based on its distance to goal. Second, when the robot is within 12m of the goal it switches to heading straight to the goal.

Racer Heavy We use an angle discretization of 5 degree width bins, $h_{thresh} = 0.15$, $\alpha = 0.1$, $\sigma_g = 70$, and $\sigma_p = 100$. Within 75m of the goal the robot would switch to heading straight for the goal. There was no linear decrease in σ_g like with Spot.

4.5.4 *Baselines*

Goal Heuristic is a simple baseline that plans the shortest path to a set of points at the edge of the costmap nearest the goal. Goal Heuristic is implemented using the Goal Conditioned Head of `LRN` but by providing a uniform distribution for $b_{filtered}$ so as to focus comparison on the image affordances from `LRN`. This approach is most similar to the common approach of assigning a (reasonably high) fixed cost to unknown space [73] and planning the shortest distance (straight line) to the goal. But by just planning to points at the edge of the costmap near the goal it removes the need to actually run search through a uniform unknown space.

NoMaD [95] is a state-of-the-art diffusion policy that predicts a trajectory from a ego-camera view and a goal image. We fine-tuned it on data from the same biome as our test sites and left Spot's internal obstacle avoidance active while forgoing the local stack, as NoMaD's designed to handle perception-to-control end-to-end. NoMaD has been shown to excel when a dense topological map is available, but our off-road experiments provide only a single goal image with no prior map. This mismatch exposes the challenges of long-horizon navigation in unmapped outdoor terrain.

Traversability + Depth Anything V2 is a combination of traversability estimation and monocular depth. The intuition is that distant traversable points should be hotspots. We first normalize their individual scores. Depth is only normalized in regions that have a non-zero traversability. We then multiply the scores to produce a heatmap similar to `LRN` and threshold the values, which are then used instead of the `LRN` hotspots.

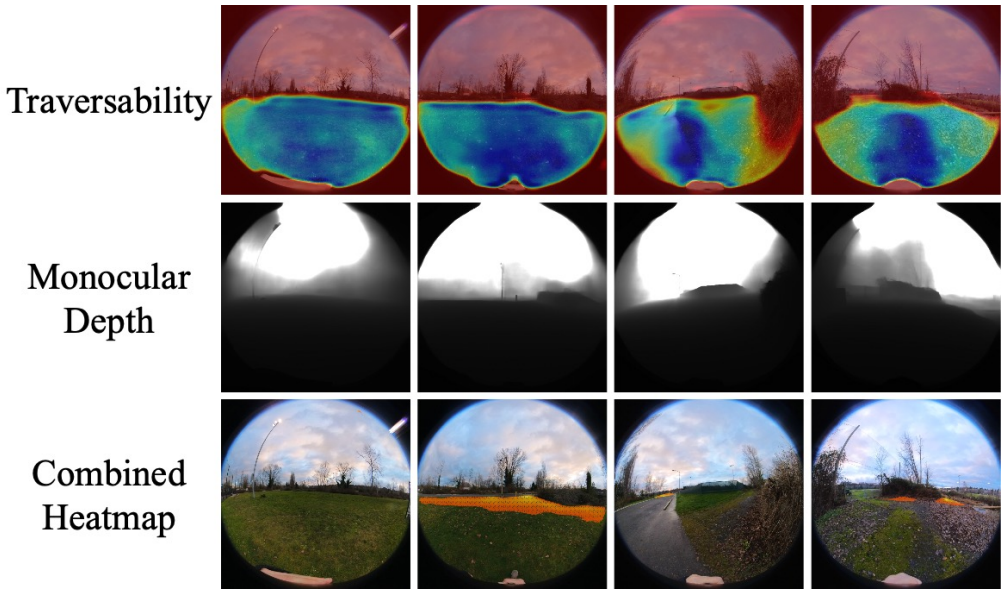


Figure 4.5: **Traversability + Depth Anything V2.** Blue indicates more traversable regions, while Red indicates less traversable areas. Similarly for depth, darker is closer and lighter is further. The resulting heatmap is thresholded to focus on hotspots, but this threshold can be overly conservative, sometimes leading to no hotspots (e.g. column 1).

The monocular depth model we used was Depth Anything V2 base model [125] which was the largest model we could run at a reasonable rate on the Orin AGX.

For Racer Heavy, we chose the V-Strong [46] traversability model, which had been trained for the California hills environment. For Spot, a V-Strong model was not available so we trained a traversability model using the same model and training as `LRN`, but instead of considering only the hotspot to be 1 in the loss, we mark the whole trajectory as traversable. To improve traversability further, we take a trick from V-Strong and expand the traversable region by making a SAM mask seeded from the robot's path.

Fig. 4.5 shows the intermediate outputs that lead to the final heatmap scores on Spot. As shown, traversability reasonably covers the accessible terrain but emphasizes regions directly in front of the robot, likely due to training trajectories heading straight out. Monocular depth gives reasonable values but becomes foggier further from the robot. Combining and thresholding the two tends to produce heatmaps that resemble `LRN` (e.g. column 3), but tend to have fluctuations in depth prediction, leading to instability in hotspot locations.

4.5.5 Real World Robot Evaluation

For Spot, we evaluated against all baselines; for Racer Heavy, only against the Goal Heuristic. All `LRN` test sites were unseen during training, though drawn from environments similar to the training set.

Spot To illustrate the challenges of using a limited range metric map, we found specific scenarios in semi-urban environments that showcase the failures practitioners would encounter with a Goal Heuristic. We tested on three courses: dump, night, and helipad as seen in Fig. 4.6.

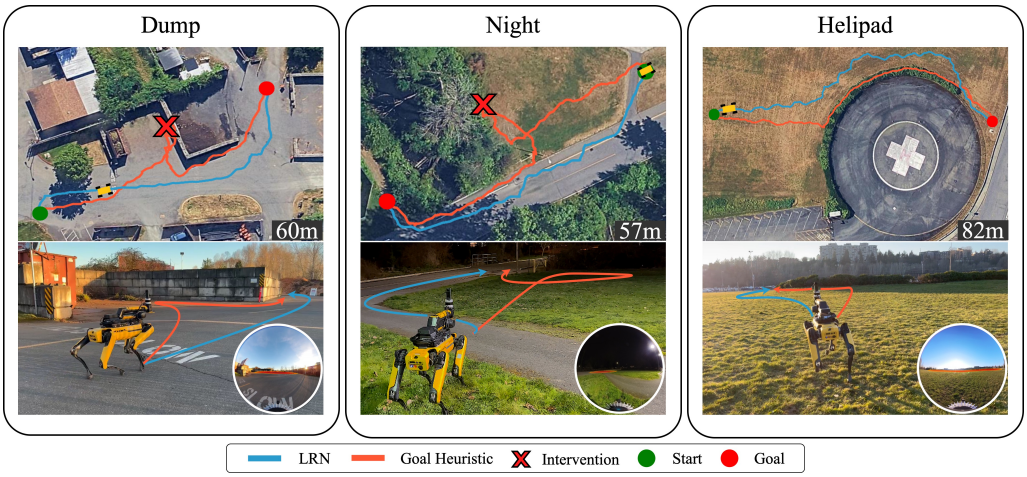


Figure 4.6: **GPS plots of LRN and Goal Heuristic Baseline on each course.** Where the Goal Heuristic blindly charges towards the goal, the LRN makes earlier decisions to avoid difficult terrain. The bottom right circular images heatmaps overlaid on image observations. In the dump and night scenario, there is high score to the side of the wall and on the sidewalk to the bridge. For the helipad course, the LRN incorrectly puts some heat on the bushes also, highlighting some sub-optimal LRN predictions.

Dump: This site has a big wall (similar to a classic bug trap) and the goal is behind it 60m from the robot's start position. The local system has good perception within its local map but the wall is outside the range of the local system until it gets close.

Night: This night course is partially lit by overhead street lights. The robot must cross a bridge and get to a point on the other side of the river 57m away. The straight line between the start and the goal is blocked by a wall of bushes and the river. Further left is a wide bridge that crosses the river.

Helipad: In this course, the robot must get to the other side of the helipad 82m away. The environment is an open field except the straight line between the robot and the goal is blocked by a wall of bushes.

Each approach was tested for five trials per course due to limited testing windows given weather and human traffic constraints. We evaluate the performance via Total Distance, Human Interventions, and Total Time. Human interventions were taken when the system was not making progress or the robot was entering a dangerous situation. Interventions were performed by facing the robot towards the goal and seeing if it will head in a reasonable direction autonomously. If not, we would walk it towards the goal until we saw a reasonable navigation plan. As the focus is on long-range navigation, we did not count interventions where local perception failed to perceive an obstacle.

Racer Heavy The Racer Heavy test was intended as a more holistic test of LRN on a full system. Thus, we provide each approach with a single waypoint 660 m away crossing three hills two of which have dense clusters of trees that should be avoided. We use similar metrics and guidelines for human interventions as Spot. Given time constraints, we were only able to run each method for one trial demonstrating the system but not fully testing it.

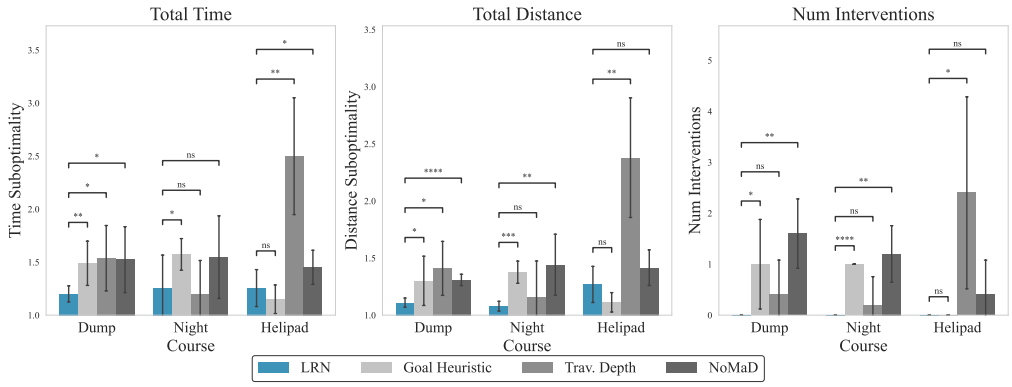


Figure 4.7: **Comparison of LRN, Goal Heuristic, Trav. Depth, and NoMaD on Spot tests.** We report average and 95 % confidence intervals for 5 real world experiments. Time and Distance suboptimality are with respect to human baseline runs. We use asterisks * to denote statistical significance : $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$, and $^{****}p < 0.0001$

4.5.6 *Offline Evaluation*

To evaluate the quality of affordance heatmaps learned offline we perform an evaluation on a human labeled test sets of 330 images for Spot and 315 images for Racer Heavy unseen during training but from the same environments. Since no other affordance heatmap predictor exists to our knowledge, we compare against Traversability + Depth Anything V2 [125].

To evaluate these methods we binarize the target heatmaps with a threshold of 0.15. We use Area Under the Receiver Operator Curve (AUROC), F1 score, Precision, Recall, False Positive Rate (FPR), and False Negative Rate (FNR). All metrics are [0, 1].

4.6 *Results*

We perform 5 tests per approach on Spot for a total of 60 trials. For ablation, we perform an additional 30 tests at the dump test site with 5 trials per heatmap threshold. Below we answer the research questions.

4.6.1 ***[Q1.]** Can the intermediate affordance representation proposed in LRN exhibit more efficient navigation capability as compared to other approaches?*

We first investigate the efficacy of LRN when compared to the baselines described in 4.5.4. For Spot, we report in Fig. 4.7 that LRN outperformed Goal Heuristic on all metrics on the Dump and Night courses and was comparable on the Helipad course. Compared to Trav. Depth and NoMaD LRN outperformed on Dump and Helipad mostly and saw competitive performance on Night course. LRN, Trav. Depth and NoMaD see a higher total distance in Helipad compared to the Goal Heuristic due to these predictive models switching directions more frequently causing wandering. Finally, we see no interventions with LRN on any course whereas all other methods incur some human interventions. Qualitative analysis shows in Fig. 4.6 that **LRN can make earlier decisions to avoid large obstacles** compared to Goal Heuristic highlighting its longer range reasoning ability (See Section 4.6.4 for more qualitative results). The GPS paths for LRN visibly are more jagged, particularly in helipad due to some switching of LRN directions which slows the robot down.

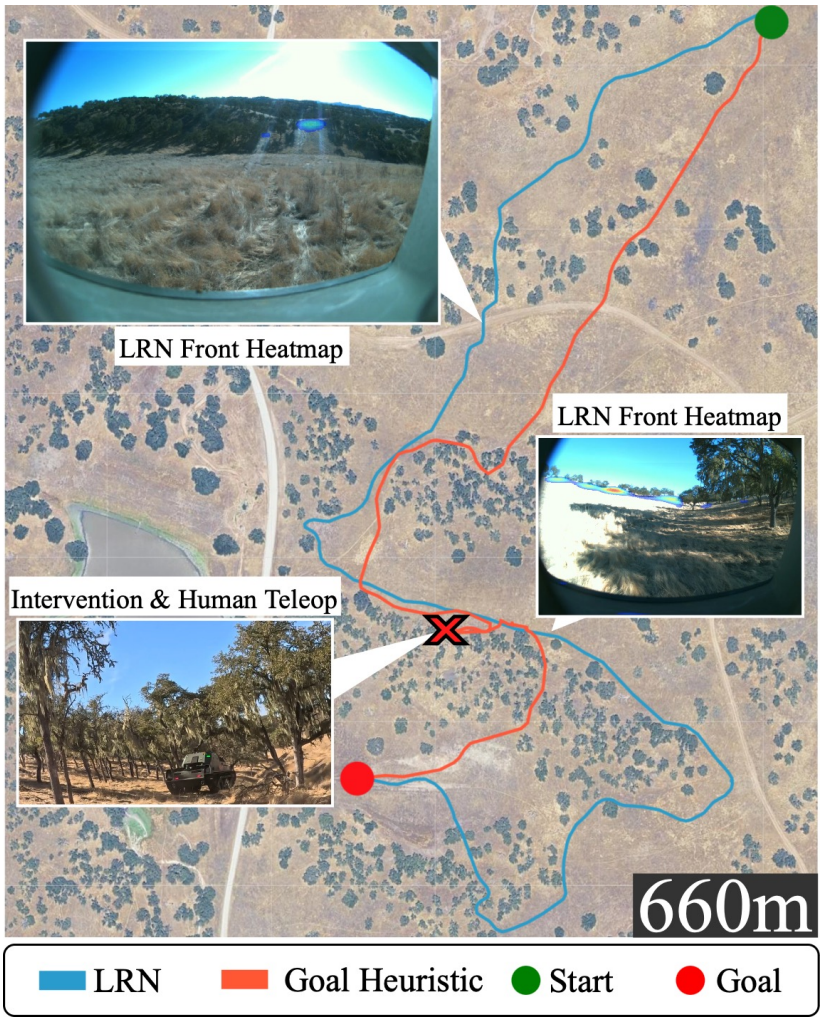


Figure 4.8: **Full-scale LRN and Goal-Heuristic demonstration**. Paths taken by baseline Goal-Heuristic and LRN systems given the same start and goal. The intervention and human teleop for the Goal-Heuristic was due to it pushing into dense trees near a ditch. The LRN avoids the dense forests on all the hills, but misses an opening toward the end and takes a slightly longer path to the goal.

Algorithm	Interventions	Avg./Max Speed (m/s)	Distance (m)	Time (s)
Goal Heuristic	1	4.02 / 8.09	**941.79**	**247.00**
LRN	**0**	**4.98 / 8.66**	1435.09	289.00

For the Racer Heavy, we report in Table. 4.8 that **LRN achieves a higher average and max speed while having zero interventions**. This is likely arrises because LRN opts for a longer but more open route around dense and difficult terrain. While this test was only of sample size one, we think this shows promise in LRN's ability to work in real navigation tasks with a full system. Qualitative analysis in Fig. 4.8 shows the Goal Heuristic baseline (orange) heads straight into the treeline on the second hill. Luckily it backs out and finds a path, only to get stuck on the next hill trapping itself. This required a 60 m human intervention from which it completed the course. In contrast, LRN (blue) found a clear opening on the second hill, avoiding the tree line. On the third hill, it opted to circumvent the entire dangerous tree line. Once past the treeline, it started heading to the goal but found greater heat Southward veering it off course but eventually finding the goal without intervention. This missed turn is likely due to our

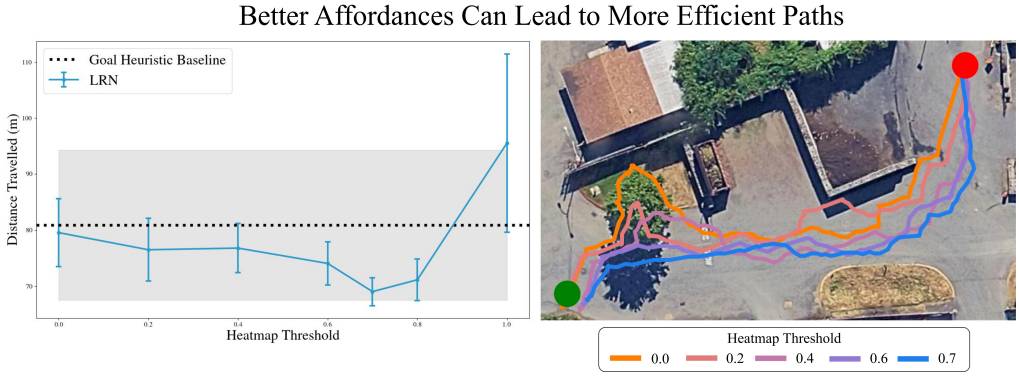


Figure 4.9: **Better affordances can lead to more efficient paths.** We modify the heatmap threshold affecting the affordance set size. Low threshold means LRN considers more potentially poor options i.e. everything is affordable. High threshold means very few or no options i.e. nothing is affordable. $h_{thresh} = 0.7$ gives the optimal affordance heatmaps resulting in shorter travel distances for LRN. We fit a line (not shown) from threshold 0.0 to 0.7 an find there is a correlation ($p < 0.05$) between affordance quality and traversal distance.

fixed σ_g, which weighted the goal the same no matter how close the robot is.

Further, we report offline test results on test sets shown in Table 4.1. We observe that LRN outperforms Trav. Depth for all metrics. We note that the performance for both Racer Heavy and Spot is better in the human labeled data. This is not surprising as any error in approximating the correct affordances induces a bias thereby impacting performance. When trained with human-expert labeled data, LRN can be viewed as equipped with oracle affordance labels, resulting in better performance.

Remark I: Through these experiments, we see LRN drives down interventions and tends to take shorter paths by seeing and avoiding dangerous terrain early suggesting it can improve overall navigation performance.

Remark II: We note that LRN's performance depends on the accuracy of the affordance model. Having access to perfect affordances e.g. in human-expert labels used for training in Racer Heavy induces lesser bias and therefore better performance. In contrast, automatic labeling of affordance labels while less burdensome can result in less accurate affordances, and therefore induce higher bias. We see this performance gap in offline evaluation shown in Table 4.1. In Spot Results, LRN sometimes exhibits switching and wandering behavior chasing various affordable frontiers. This is in part due to not modeling the correct affordances. We note wandering can also be the cause of improper tuning of the goal conditioned head or the goal conditioned head not fully capturing the true goal-conditioned cost.

4.6.2 *[Q2.] Considering the connection between the quality of the affordance model against system performance, do better affordances lead to more efficient paths?*

To evaluate the connection between affordance quality and navigation performance, we perform an ablation study on the Dump course as shown in Fig. 4.9. To vary the affordance heatmap quality, we adjust the heatmap threshold h_{thresh} in the range 0 to 1.0. At 0, much of the environment is predicted as affordable, causing the robot to take a more direct path to the goal–often getting stuck in local minima like the Goal Heuristic. At 1.0, almost none of the environment is deemed affordable and the robot switches between heading directly towards the goal and a random hotspots that appear infrequently. Between those extremes, we see how h_{thresh} best tuned for the

System	Metric	LRN Auto	LRN Hand	Trav. Depth
Racer Heavy	AUROC ↑	0.63	**0.84**	0.56
	F1 ↑	0.11	**0.52**	0.09
	Prec. ↑	0.08	**0.51**	0.07
	Rec. ↑	0.17	**0.52**	0.13
	FPR ↓	0.03	**0.01**	0.03
	FNR ↓	0.83	**0.48**	0.87
Spot	AUROC ↑	**0.93**	0.77	0.61
	F1 ↑	0.10	**0.32**	0.14
	Prec. ↑	0.06	**0.30**	0.14
	Rec. ↑	**0.35**	**0.35**	0.13
	FPR ↓	0.01	**0.0**	**0.0**
	FNR ↓	**0.65**	**0.65**	0.87

Table 4.1: **Classification metrics for heatmap backbone on test set.** Prec. and Rec. stand for Precision and Recall. The F1/Prec/Rec/FPR/FNR/ are from the highest scoring heatmap thresholds for each method: Trav. Depth and LRN.

system directly translates to the best navigation performance.

Remark III: By adjusting the heatmap quality via h_{thresh}, we see a correlation between improved affordances and improved navigation performance, indicating that there is an intermediate affordance value of affordance set size, that drives most gains when compared to the Goal Heuristic. Considering almost all directions as affordable or few affordable both can hurt the system which is intuitive. But we note, better affordances may not always lead to shorter paths as LRN is a heuristic that cannot predict what the environment will be like beyond view.

4.6.3 *[Q3.] How does auto-labeling versus human labeling affect affordance quality?*

We report offline test results in Table 4.1. We note that the performance for both Racer Heavy and Spot is better in the human labeled data. This is not surprising, since auto-labeling errors bias affordance estimates and degrade performance. When trained with human-expert labeled data, LRN can be viewed as equipped with oracle affordance labels. "Nonetheless, both hand- and auto-labeled LRN outperform Trav. Depth across all metrics, suggesting auto labels still provide a useful learning signal. This is supported by Spot test where LRN, trained only with auto-labels, outperformed other methods.

Remark III: We note that LRN's performance depends on the accuracy of the affordance model. Having access to perfect affordances e.g. in human-expert labels used for training in Racer Heavy induces less bias and therefore better performance.

In contrast, while auto-labeling reduces effort, it can yield less accurate affordances—resulting in higher bias, as seen offline and in Spot's wandering behavior. That being said, empirical Spot results suggest auto-labels still provide sufficient signal to improve navigation performance.

4.6.4 *Additional Qualitative Results*

Racer Heavy Heatmaps A sample of qualitative heatmap results can be found in Fig. 4.10. Traversability + Depth Anything V2 has varying levels of performance. In

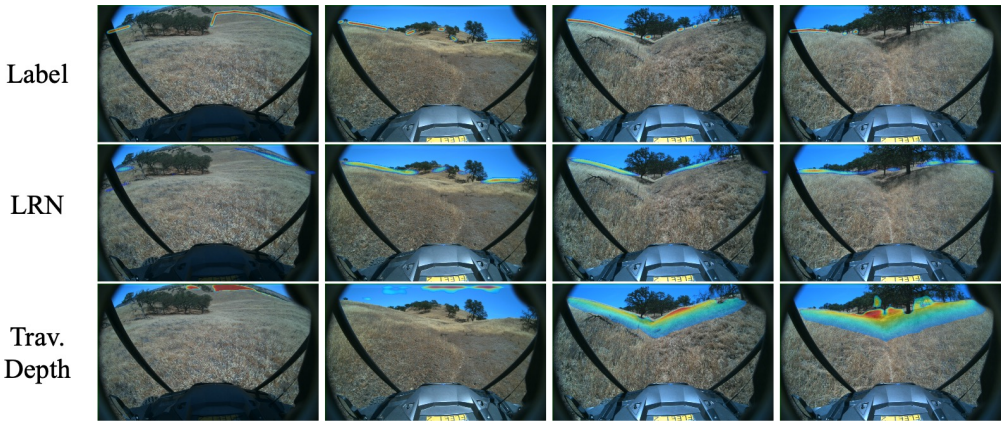


Figure 4.10: **Racer Heavy heatmap predictions** compared to human-labeled heatmaps on the test set.

the leftmost image, it finds distant hills traversable and thus predicts them as a high score, not considering the uncertainty of getting to the hills. Right of that, it predicts sky as traversable and distant. This happens on and off and is due to fluctuations in depth predictions from Depth Anything V2. In the next two right images, it gets close to the correct hotspots but has no reasoning for whether the robot can continue from that point, thus marking paths leading into dense trees as traversable.

`LRN`, in comparison, gets much closer to the human labels, identifying key openings between trees. While it mostly gets the correct hotspots, it tends to smooth the heat between them more than the true labels, resulting in some heat on undesirable areas.

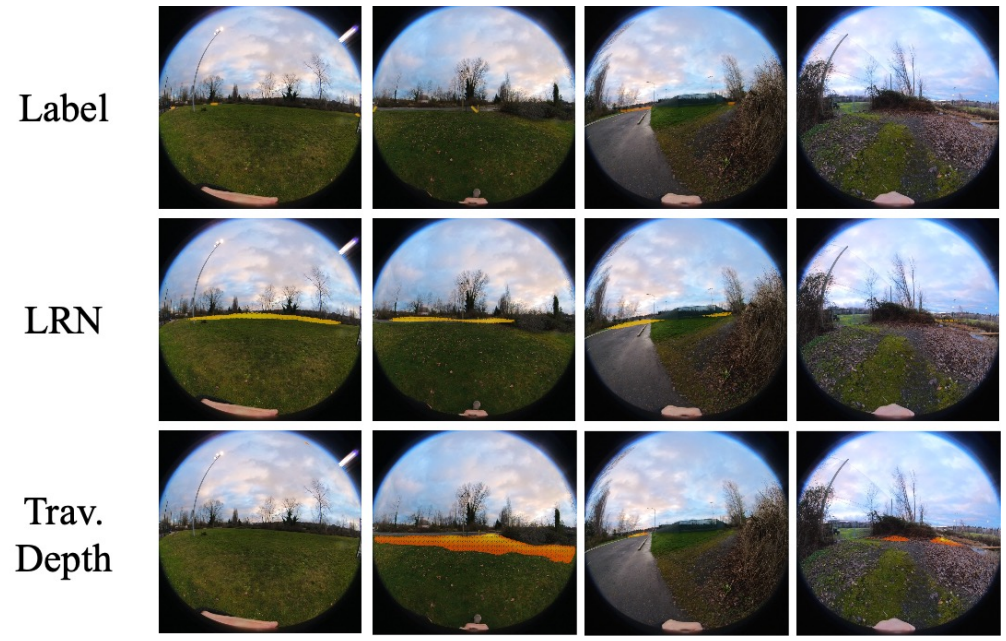


Figure 4.11: **Spot heatmap predictions** compared to human-labeled heatmaps on the test set.

Spot Heatmaps Qualitative heatmap results are presented in Fig. 4.11. As shown, Traversability + Depth Anything V2 is very sensitive to fluctuations in depth predic-

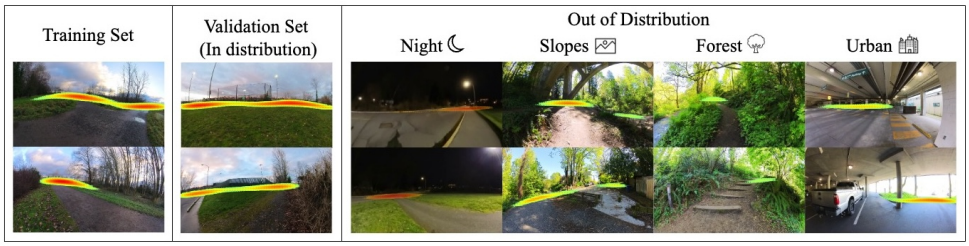


Figure 4.12: **OOD predicted heatmap examples.** Shown are images with a 0.5 threshold on heatmaps (nominal 0.7) to allow for lower-confident OOD predictions.

tion, sometimes giving no hot spots in the heatmap, whereas LRN tended to be more stable. LRN also seemed overly optimistic compared to human labels, which we think contributes to some of the switching behavior in real-world tests.

OOD Qualitative Results Although our method enables fast re-training in a new environment, we expect the vision-backbone model to generalize somewhat to OOD conditions. Fig. 4.12 shows a few examples in diverse environments with various biomes (urban/forest), lighting conditions (night/day), and slopes (hilly/flat). The model was trained on data from walking around a park, which had some overcast/sunny skies, flat ground, and brown winter vegetation. Though OOD predictions are lower confidence, they still identify promising frontiers-—highlighting the strength of vision foundation backbones.

Spot Qualitative Runs Fig. 4.13 shows sample paths each approach took on all courses. As shown, there were multiple interventions for the baselines because the robot got off course and needed to be corrected. We also see variations in performance of Traversability + Depth Anything V2 and NoMaD across courses. For example, Traversability + Depth Anything V2 does quite well in the Night course but on Helipad incurs a lot of wandering due to the more open environment.

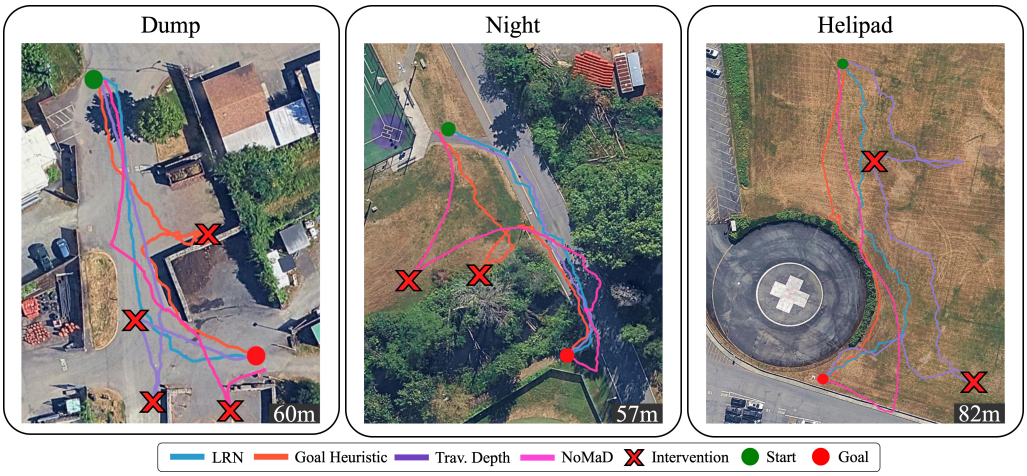


Figure 4.13: **GPS plots of all approaches on each course.** Many of the baselines incurred interventions for going off course and exhibit various degrees of wandering

4.7 Limitations

While LRN has shown better overall long-range navigation, our method is not without its limitations. Here, we discuss key failure modes.

First, LRN does not reason about depth explicitly. Without depth, we are implicitly assuming that the angular distance to goal from an LRN hotspot is a sufficient proxy for distance to goal. This assumption can break when two or more hotspots are equal angular distance from the goal heading, but in reality, one is much closer to the goal. This appears as occasional wandering on the Helipad and Racer Heavy courses due to open environments with many hotspots in LRN predictions. That said, LRN recovered by eventually finding more direct hotspots and reaching the goal without intervention. Beyond incorporating depth, this issue can be addressed by adding a detector for wandering behavior and reducing σ_p, encouraging sticking to one decision or reverting to Goal Heuristic until fluctuations in LRN stabilize. A good signal for this is non-monotonic erratic fluctuations in distance to goal, which may not always be decreasing (e.g. going further around an obstacle), but should change smoothly.

Second, LRN can exhibit switching behavior due to fluctuating heat scores. Small fluctuations in score between two very different directions cause the robot to switch back and forth. This problem motivated the EMA filter and previous heading gaussian score, but we found it does not completely alleviate the issue, and more exploration is needed. We would like to explore learning the goal conditioned head with history to see if it can learn to maintain consistent headings.

Third, while some heatmaps seemed reasonable in online tests, we noticed more optimism from the Spot model where it puts heat on obstacles near an opening. We attribute this to the automated labels, which some have small tracking errors putting heat on the edges of openings. Future work on reducing tracking error or filtering bad labels could improve performance on this front.

Finally, LRN is a heuristic for exploring unknown space. While we show it can be an improved heuristic over other methods, all heuristic frontier approaches suffer from not truly knowing the whole environment. Thus, the LRN cannot guarantee improved performance because a frontier that looks good from one perspective may actually lead down an unseen bad path. Incorporating history into the LRN could help, so if the robot reached a dead end it could remember a previous hotspot and backtrack to that position.

4.8 Research Questions

RQ3 Using only on board sensing, is it possible to traverse a similar path as one planned using privileged global information?

Early on in this work, we tried comparing to a global planner that had access to geospatial information. We found the plans were suboptimal compared to human paths due to inaccuracies in the geospatial information. Thus, in Fig. 4.7 we opted to compare directly to human paths from people who had seen the whole course. Given that, we

see it is possible to traverse paths close to human performance on simple courses (distance suboptimality close to 1.0) and better than most other methods. In particular, LRN avoids interventions much like a human would by avoiding dangerous terrain, most notably in Fig. 4.8.

As noted in Section 4.7, LRN is a heuristic for exploring unknown space. And all heuristic frontier approaches suffer from not truly knowing the whole environment. Thus LRN cannot guarantee improved performance because a frontier that looks good from one perspective may actually lead down an unseen bad path. This is true for all navigation strategies that rely on partial information, so while it is possible to traverse paths similar to one's with privileged information, it cannot be guaranteed in large more complex environments.

RQ4 Where is long-range navigation not useful?

The intuitive answer is that long-range navigation is not useful when the robot's visibility is limited. While somewhat true, through this work we find there is more to the story. First, in tighter spaces where there is little sensor data beyond the local map, like some of the forest on the Racer Heavy test, LRN finds reasonable openings for the robot to choose between. While these openings are visible in the costmap, the additional signal for planning to head to one in particular does not seem to hurt performance with the robot reaching the goal just fine. It is possible this could improve robustness as LRN serves as second form of perception, finding openings in tight spaces. Second, in very open spaces, LRN experiences wandering, seeing many possible directions at once and struggling to prioritize. This is particularly unhelpful when the goal is visible and the robot should just head directly toward it. We note these scenarios are specific to LRN; incorporating depth or other methods may help prevent wandering. Additionally, adding some check for if the goal is visible based on its distance and coarse depth.

5
Fast Adaptation From Videos

This chapter addresses RQ5: how to rapidly adapt with new off-robot experiences to make better decisions under uncertainty. As shown in Chapter 3, uncertainty reasoning is important for robust performance during deployment. However, reasoning about uncertainty at deployment addresses only half the challenge. The other important factor is reducing uncertainty. This chapter explores how to rapidly retrain perception and planning models with new experiences to achieve accurate predictions in novel environments.

5.1 Introduction

The traditional pipeline for building a learned robotic system involves collecting a large data corpus of data, train perception models, tune controllers, and deploy. For this to work, practitioners assume the training data comes from the same distribution as the deployment environments. In the off-road setting, this can be difficult in practice. Off-road environments span diverse biomes, but no comprehensive driving dataset covers them all—-and assembling one would be prohibitively expensive.

An alternative to zero-shot deployment is few-shot adaptation [72], where models receive a small amount of deployment-environment data to fine-tune a base model. Practically, this works well if the few-shot data is readily available or easy to collect. Labeling semantic classes for fine-tuning classifiers can be arduous limiting how fast the system can adapt in a new environment.

Multiple recent works have shown promise learning traversability models (an upper bound on where the robot can and can't go) from just the robot's odometry and proprioceptive sensors (no labeling) [92; 32]. These approaches are great for when the robot is in the deployment environment, but are limited to on-robot data. But off-road robots, like the RACER Heavy can be large and difficult to transport making it expensive to collect data. This motivates us to ask the following research question:

How can we quickly train off-road robots without robot specific data?

We make two **key observations**: First, terrain affordances and feasible trajectories are largely predictable from visual cues of the scene. Vision alone can serve as a prior on traversability and possible trajectories, while a handful of proprioceptive signals are

sufficient to adapt that prior to the physical limits of a new platform. Learning from visual cues removes the requirement of on-robot data, enabling practitioners to adapt their system from videos taken in the environment.

Second, trajectories from ego-centric (first person view) videos naturally embed useful navigation affordances. Where the human/car/robot went implies what types of terrain are feasible and how to move in that terrain. Video trajectories can be obtained through monocular slam, or point tracking, both of which we explore in this work. This leads to our **key insight**: by distilling the implicit navigation cues contained in egocentric videos into visual navigation priors, and then fine-tuning with a handful of platform-specific proprioceptive signals, we can deploy off-road robots in novel environments with almost no on-robot data collection—dramatically shortening the adaptation cycle.

To this end, we propose a video-based pre-training framework that employs off-the-shelf point-tracking to recover egocentric trajectories from first-person footage. We show that the resulting trajectories are similar to ground-truth odometry and yield comparable performance when used to train downstream traversability and trajectory prediction models.

5.2 *Problem Statement*

Let $V = \{I_t\}_{t=1}^{T}$ be a first-person video composed of T RGB frames $I_t \in \mathbb{R}^{H\times W\times 3}$. Our objective is to create a trajectory estimator

$$f:\ V \longrightarrow \hat{\mathbf{P}} = \{\hat{\mathbf{p}}_t\}_{t=1}^{T} \qquad \mathbf{p}_t = (u_t, v_t),$$

that estimates the 2-D image path $\hat{\mathbf{P}}$ of the filmer. The estimate $\hat{\mathbf{P}}$ should act as a drop-in replacement for the ground-truth path obtained by projecting accurate 3-D odometry into the image.

$$\mathbf{P}^{\star} = \{\mathbf{p}_t^{\star}\}_{t=1}^{T}, \qquad \mathbf{p}_t^{\star} = (u_t^{\star}, v_t^{\star}),$$

Performance is measured only through downstream tasks. Let $\mathcal{T}$ denote a collection of tasks (e.g., traversability estimation, future-path prediction); each task $t \in \mathcal{T}$ provides a learner $A_t(\cdot)$ that maps a trajectory sequence to a hypothesis h, and a loss function $\mathcal{L}_t(h, x)$ evaluating the hypothesis on samples $x \sim \mathcal{D}_t$.

We seek an estimator f that minimises the average discrepancy in expected task loss between hypotheses trained with ground-truth trajectories and those trained with video-derived trajectories:

$$\arg\min_{f}\ \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\left|\mathbb{E}_{x\sim\mathcal{D}_t}\big[\mathcal{L}_t(A_t(\mathbf{P}^{\star}), x) - \mathcal{L}_t\big(A_t\big(f(V)\big), x\big)\big]\right|.$$

Note, this objective serves to provide clarity but we will not directly optimize it because f is a algorithm not a function.

5.3 Related Work

5.3.1 Point Tracking

Point tracking follows 2D image points as the scene changes due to camera and object motion. There are a variety of point trackers available for use [47; 23; 24; 113], and for this work we leverage CoTracker [47], which tracks grids of points, enabling robustness to individual point occlusions. Tracking is done from a learned model that finds correspondences between consecutive frames. Point tracking has advanced greatly: it now generalizes across scenes and track through occlusions. It has seen applications in robot manipulation via imitation learning [114; 9; 120] and sim to real transfer [130] but notably few works have used point tracking for navigation. The likely reason is point tracking can only track effectively for short clips, and once the points are out of frame they are lost. This work presents a way to re-initialize tracking to overcome these challenges.

5.3.2 Monocular Slam

An alternative to point tracking for extracting trajectories is monocular SLAM [103; 14; 117; 64]. Monocular SLAM methods go beyond point tracking to reconstruct scenes and predict camera poses in 3D. They work by training neural networks to find correspondences between frames and estimate depth from the monocular image. Because they are a full SLAM approach, they benefit from further global optimization and loop closure, which can mitigate errors from individual predictions. While promising, many of these approaches are trained on indoor or human-centric environments and struggle with outdoor terrain. We compare to the leading approach MASt3R-SLAM [64] as an alternative tracker.

5.3.3 Self-supervised Traversability Estimation

Self-supervised learning has emerged as a promising approach to improve off-road navigation by estimating traversability. Near to far learning, discussed in Section 4.3.1, is a self-supervised approach to predicting the traversability of terrain further from the robot using high-confidence predictions near the robot. Similarly, multiple self-supervised approaches have used onboard sensing (usually IMU) to train a learner to predict terrain roughness or traversability [35; 115; 118; 45; 80; 46; 92] using RGB/depth as input. In particular, methods like SALON [92] leverage foundation models to adapt with only seconds of data. While these methods focus on adaptation with robot data, we are interested in what adaptation can be achieved off-robot data.

5.3.4 Learning from Videos

Learning from videos has been a growing approach in robotic manipulation [50], but less attention has been given to navigation. Notably, Kumar et al. [52] demonstrates learning navigation subroutines from ego-centric videos. From ego-centric videos they

extract routines like "turn left" via learning a inverse mapping from consecutive video frames to latent actions. Our approach differs in that we are extracting interpretable trajectories and using them for various downstream tasks.

5.4 *Extracting Trajectories from Videos*

5.4.1 *Tracking*

To extract 2D trajectories from videos, we leverage the strong tracking abilities of CoTracker. CoTracker differs from other trackers in that it tracks a grid of points simultaneously. The motivation for tracking a grid is that many points are correlated in how they move. This is particularly true when considering camera motion where all points in a static scene will move together. That being said, CoTracker does not assume static scenes and shows multiple examples of tracking moving objects and the background.

CoTracker works by tracking a fixed grid of points, noting when they become occluded. It is not possible to add points to the scene while tracking. This presents two problems. First, how do we track points that go behind the camera as it moves forward? Second, how do we add new trajectory points as the camera moves? The first issue is easy to solve, we simply track the video *in reverse*. As the camera moves backwards points will can be tracked until they become occluded. The second issue can be solved by re-initializing the tracker whenever we want to add a new point. To avoid losing prior tracks, we combine all existing points with the new point into one grid, then re-initialize the tracker. This works provided we do not re-initialize points that have significantly drifted or been occluded, since those would latch onto incorrect scene regions. For a full description of the algorithm see Algorithm 3. Finally, we add a point every few frames at the ground-center of the image (near the bottom) to capture points right in from of the filmer. For best performance, a wide vertical FOV or a camera pointing slightly downward is recommended.

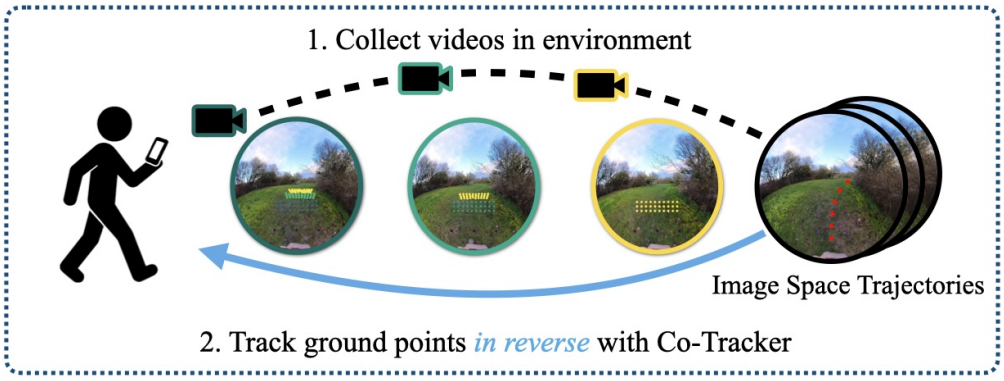


Figure 5.1: **Extracting Trajectories from Videos** We collect egocentric walking videos and track ground points in reverse to extract trajectories for training

5.4.2 *Additional Tracking Improvements*

To prevent drift, we set a low occlusion threshold and, as recommended in CoTracker, track a local grid around each point in addition to the full-image grid. Since we are tracking local square grids that get reinitialized we resize the grids based on their final width in the previous cycle (See Fig. 5.1).

For cleaner results, we discard trajectories containing gaps longer than λ frames. This prevents the occasional singular distant point from getting locked on an incorrect position. We also remove duplicate points that lie very close, since overlap increases with depth.

We post-process the tracked grids to get a cleaner set of trajectories with a fixed number of points. First, we compute convex hulls between each of the grids. Then, we compute the centerline along each hull connecting hulls at their midpoints. From the centerline, we resample a fixed number (20 in our experiments) of points along the trajectory. This process helps bridge occlusions and yields smoother trajectories compared to raw tracked points.

Algorithm 3 `Tracking`

Require: Video V, interval k, new point p_{new}, grid width w, visibility threshold q, distance threshold δ, min occluded λ

$V_r \leftarrow \texttt{reverse}(V)$
$P \leftarrow \{\}$ ▷ Tracked points
$W \leftarrow \{\}$ ▷ Grid widths
for all $I_i \in V_r$ **do**
  if $i \bmod k = 0$ **then** ▷ Periodically add new point
    $P \leftarrow P \cup \{p_{\text{new}}\}$
    $W \leftarrow W \cup \{w\}$
    $G \leftarrow \texttt{Make\_Grids}(P, W)$
  $O,\ P,\ G \leftarrow \texttt{Track}(P, G, I_i)$
  $m \leftarrow \lambda$
  for all $(o_j, p_j, g_j, w_j) \in (O, P, G, W)$ **do** ▷ Occlusion filtering
    if $o_j < q$ **then**
      $P \leftarrow P \setminus \{p_j\}$
      $G \leftarrow G \setminus \{g_j\}$
      $W \leftarrow W \setminus \{w_j\}$
      $m \leftarrow m - 1$
      if $m = 0$ **then** ▷ λ occlusions reached, clear remainder
        $P \leftarrow P \setminus \{\, p_\ell \mid \ell > j \,\}$
        $G \leftarrow G \setminus \{\, g_\ell \mid \ell > j \,\}$
        $W \leftarrow W \setminus \{\, w_\ell \mid \ell > j \,\}$
        `Break`
  $P, G, W \leftarrow \texttt{RemoveDuplicates}(P, G, W, \delta)$ ▷ Prune duplicates
return $P,\ G$

Algorithm 4 `RemoveDuplicates`

Require: Point set P, grid set G, width set W, distance threshold δ

```
for j ← 1 to |P| − 1 do
    for ℓ ← j + 1 to |P| do
        if ‖p_j − p_ℓ‖_2 < δ then
            P ← P \ {p_ℓ}
            G ← G \ {g_ℓ}
            W ← W \ {w_ℓ}
return P, G, W
```

5.5 *Experimental Setup*

To understand how effective the proposed tracker is, we seek to answer the following two questions.

[Q1.] How similar are tracked trajectories compared to ground truth? (See Sec. 5.6.1)

[Q2.] Does the error in tracking make a statistically meaningful difference in down stream task performance? (See Sec. 5.6.2)

To answer these questions, we run the tracker on the RELLIS-3D [44] off-road dataset, which has reliable fused odometry to compare against. RELLIS-3D additionally has ground truth semantic segmentation of the front-facing camera.

Tracked trajectories and projected odometry may not have the same length or number of points. We align the two by cropping longer odometry trajectories to match the length of tracked trajectories. Then we evenly resample the trajectory so it has the same number of points as the tracked trajectory. This implies we evaluate only the quality of successful tracks, not failure cases. As Tracking is used for downstream applications, filtering out failures is easy to do with simple rules.

For Q1, we evaluate tracked trajectories against projected odometry via both the continuous Fréchet and discrete L2 distance. While L2 is common, Fréchet distance better accounts for differences in timing. For example, if the tracked trajectory has more spread out points than the odometry, even after syncing and resampling, L2 will penalize that while Fréchet allows the trajectories to move at different speeds.

For Q2, we train models for two relevant tasks to off-road autonomy: image traversability prediction, and image trajectory prediction. We summarize those methods below.

5.5.1 *Traversability Prediction*

We train image based traversability via V-STRONG [46] which learns traversability via combining visual foundation model embeddings, contrastive loss, and pseudo-labeling via SAM [48]. V-STRONG uses only non-occluded points along the camera path to predict traversability via similarity to traversed pixels. We train separate V-STRONG models on tracked vs. ground-truth trajectories and evaluate their binary classifications (traversable vs. not) on RELLIS-3D. Traversability is predicted on a [0,

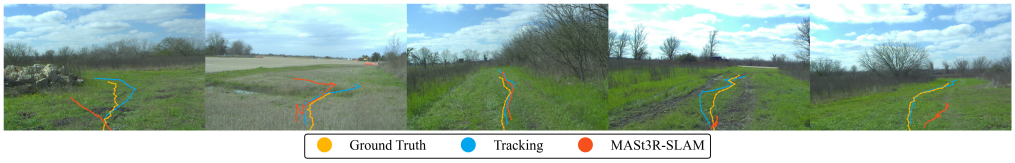


Figure 5.2: **Qualitative Tracking Comparison.** Visual comparison of Tracking and MASt3R-SLAM on the RELLIS-3D dataset. As shown, Tracking may be slightly off ground truth but get the general shape right, whereas MASt3R-SLAM struggles to track in most cases in challenging outdoor environments.

1] scale, and we provide ROC curves for various thresholds along with metrics for the best performing thresholds. For training, we use 9,884 images and evaluated on a hold out 899 images. We measure performance with ROC curves and for each of the best performing thresholds we additionally measure accuracy, precision, recall, and F1 along with statistical tests.

5.5.2 *Trajectaory Prediction*

We opt to train a Diffusion Policy [16] for trajectory prediction as it has been used for trajectory prediction in other domains [57; 5]. The policy is given a goal point and creates a plan from a fixed point just ahead of the robot. We evaluate Diffusion Policy on the RELLIS-3D dataset and compare the best/average trajectory prediction error (Fréchet/L2) compared to the ground truth trajectory. Further, we leverage RELLIS-3D semantic classes to evaluate the proportion of the trajectory in a non-traversable part of the image. As with traversability, we compare training Diffusion Policy with tracked and ground truth trajectories on the same train/test set.

5.6 *Results*

We present results comparing Tracking and MASt3R-SLAM to ground truth for tracking performance, traversability estimation, and trajectory prediction. MASt3R-SLAM experienced poor tracking outdoors, often yielding no points to project. Because of this poor performance, MASt3R-SLAM was unusable for training downstream models.

Algorithm	L_2	Fréchet
Tracking (full)	$\mathbf{0.062 \pm 0.001}$	$\mathbf{0.091 \pm 0.001}$
Tracking (602)	$\mathbf{0.064 \pm 0.003}$	$\mathbf{0.100 \pm 0.006}$
MASt3R-SLAM	0.163 ± 0.007	0.23 ± 0.010

Table 5.1: **Tracking metrics.** We perform a difference of means test between Tracking (602) and MASt3R-SLAM and find $p < 0.00001$ showing statistically different results.

5.6.1 ***[Q1.]*** *How similar are tracked trajectories compared to ground truth?*

In Table 5.1 we can see the tracking metrics. Raw L2 and Fréchet were scaled by image width, so Tracking's (full) average Fréchet error is less than 10% of the image for the full dataset. Qualitatively in Fig 5.2 that Tracking produces similar trajectories to ground truth but that the tracked trajectories are more smooth than ground truth. We attribute this to the skid-steer vehicle platform used in data collection had a lot of turn in place turns which odometry reflects but Tracking will smooth out. Tracking (602) is Tracking evaluated only on the 602 images MASt3R-SLAM had a solution for. Tracking (602) also outperforms ($p < 0.00001$) MASt3R-SLAM on all metrics, suggesting Tracking is more

reliable for extracting trajectories from videos. We attribute MASt3R-SLAM's poor performance to noisy outdoor scenes with less flat planes and more organic shapes (e.g. small sticks) that can induce visual SLAM error.

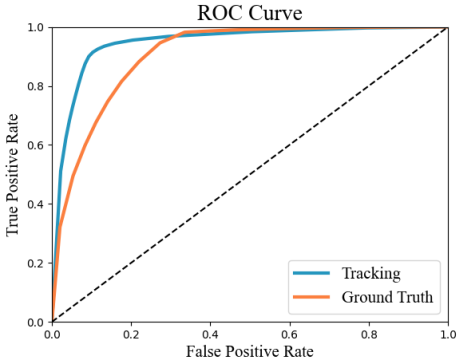


Figure 5.3: **ROC curve for Tracking and Ground Truth.** We see Tracking performs slightly better achieving a higher TPR without a high FPR.

5.6.2 *[Q2.] Does the error in tracking make a statistically meaningful difference in down stream task performance?*

We compare against two relevant downstream objectives: traversability estimation and trajectory prediction. Traversability prediction results shown in Table 5.2 and the receiver operator curve in Fig. 5.3 suggests Tracking surprisingly outperforms training on Ground Truth trajectories on all but Recall. We also perform a moving block bootstrap for each metric to test that they have a difference less than 0.05 of the score. These results did not show significance suggesting Tracking results are not effectively equivalent to Ground Truth. Qualitatively in Fig. 5.4 we also see Tracking is slightly more confident in its labels and produces more crisp separation between traversable and non-traversable. So why does Tracking seem to perform better? We believe this is due to Tracking errors causing wider and less direct paths. These wider paths may cover a wider visual diversity of terrain signaling more of the environment is traversable. Combined with the wide grass section prevalent in RELLIS-3D this may actually help the classifier better distinguish grass from non-grass terrain but would have the opposite effect in tighter environments.

Algorithm	Accuracy	Precision	Recall	F1
Tracking	$\mathbf{0.906 \pm 0.0001}$	$\mathbf{0.912 \pm 0.001}$	0.925 ± 0.0001	$\mathbf{0.919 \pm 0.001}$
Ground Truth	0.852 ± 0.0001	0.822 ± 0.001	$\mathbf{0.946 \pm 0.0001}$	0.880 ± 0.0001

Table 5.2: **Traversability Classification Metrics**. Metrics are from the best performing threshold for each approach. We perform a TOST test of equivalence with a moving block boostrap with block size of 10 and 100 bootstrap samples for each metric. We find no statistical significance suggesting these results are not statistically similar.

For trajectory prediction, we see in Table 5.3 Ground Truth statistically outperforms tracking on everything except L2 (not significant). This is not surprising as training on Ground Truth should make the model at better at predicting Ground Truth trajectories in evaluation. But while we see Tracking performs worse, it does not perform that much worse. In particular Fréchet error is 0.06 of the image whereas Ground Truth recieved 0.04. And the proportion of the trajectories in collision is low for both. Looking at Fig. 5.5 we see similar trajectories for the predictions suggesting Tracking provides a useful training signal.

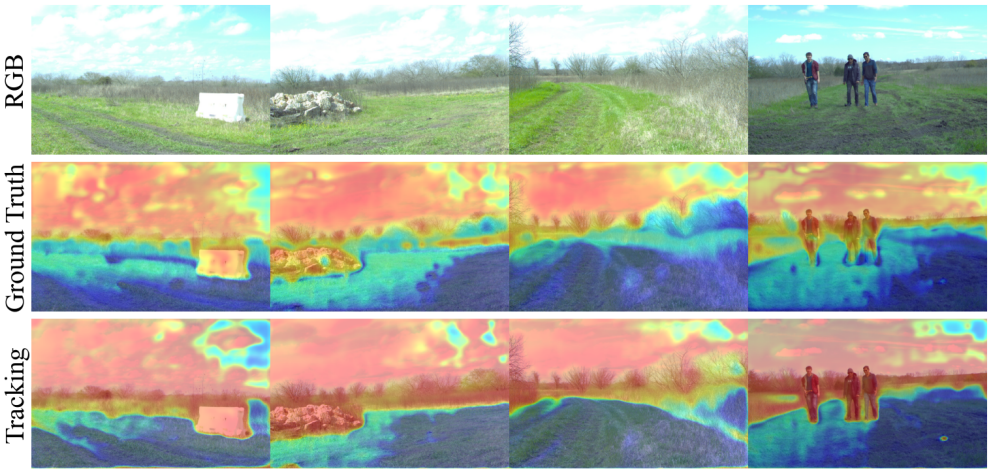


Figure 5.4: **Qualitative Traversability Predictions.** Red indicates less traversable while blue indicates more traversable. As shown Tracking seems to have cleaner separation between classes and more confidence.

Training Data	L_2	Fréchet	Violations
Tracking	0.239 ± 0.005	0.066 ± 0.017	0.002 ± 0.001
Ground Truth	$\mathbf{0.118 \pm 0.0032}$	$\mathbf{0.040 \pm 0.001}$	$\mathbf{0.001 \pm 0.001}$

Table 5.3: **Trajectory Prediction metrics compared to ground truth**. We perform a TOST equivalence test and find statistically different results for all but L2.

5.7 *Limitations*

While our trajectory extraction approach shows a promising way to scalably train navigation policies from easy-to-collect videos, it is not without its limitations.

First, this method by design only extracts 2D trajectories whereas a robot would operate in 3D. Consequently, any trajectory predictor must be post-trained on 3D data to make accurate 3D predictions.. This could achieved by pre-training on (u, v, d) where d (depth) is fixed at pre-training and post training on inversely projected 3D trajectories (pixel space) with real depth. By post-training on pixel (u, v) and depth it enables efficient transfer of the prior compared to predicting in 3D, which is very different than pixel space.

Second, this work has yet to explore 3D post-training in depth and deploy on real hardware. For these tests we would want to compare models trained purely on the robot-specific 3D trajectories with ones given access to pre-training before fine tuning. Our hypothesis is that additional large-scale pre-training will benefit both trajectory prediction and full-system performance.

Third, our approach makes specific assumptions about the video format to work well. It assumes the video is first-person with a good view of the ground in front of the camera. We also assume the camera's mounting remains roughly fixed relative to the filmer's body. These assumptions limit the types of videos our method can be applied on. Future work could add ground-plane detection to avoid obstacle points and add a classifier to distinguish forward motion from rotation. We find cameras with wide vertical and horizontal FOV—-ideally tilted downward—-work best to view the ground and handle in-place turns. Interestingly, monocular SLAM does not have these

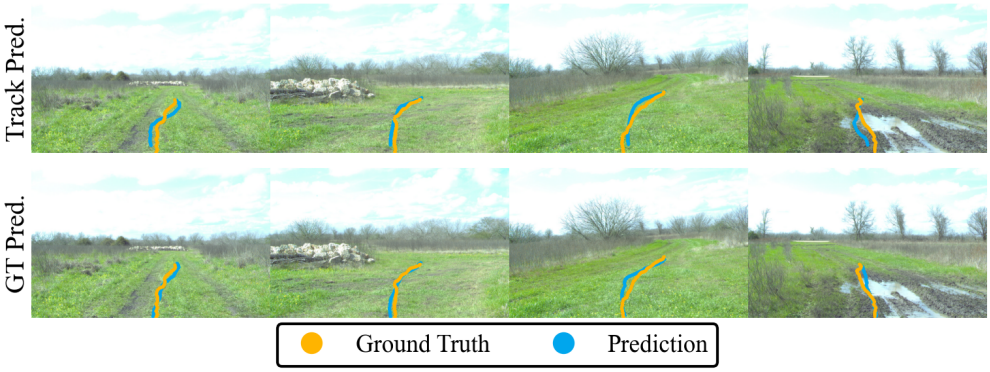


Figure 5.5: **Predicted Trajectories.** We see both prediction methods produce trajectories close to ground truth suggesting Tracking can provide useful signal for trajectory prediction.

limitations but struggles in outdoor environments. Future work could investigate a hybrid approach leveraging the strengths of both methods.

Fourth, our trajectory extractor cannot reason through occlusions by default. Tracking will crop the trajectory at the point of occlusion. This prevents learning trajectories that go around obstacles where free space exists on both sides. Instead, it learns to go up to the point of occlusion. Future work should investigate if 3D post-training on full trajectories mitigates this issue.

5.8 Research Questions

RQ5 Can off-robot experience improve decision making under uncertainty?

In this work, we explore if it is possible to quickly train perception and planning models using easy to collect unlabeled video data. While the results presented are limited to offline evaluation, models trained on data from our Tracking algorithm (run only on videos) show minimal performance gap compared to using robot odometry. This suggests off-robot experience from unlabeled ego-centric videos can in fact improve model performance and thus produce models with less uncertainty. To fully address this question, experiments evaluating if perception models can be fine-tuned for new environments with tracked video data should be performed with a real robot. We also note traversability estimation and trajectory prediction do not capture all of the tasks a robot may be trained and/or uncertain over. For other tasks (e.g. elevation prediction), videos may not be appropriate.

6
Conclusion

This dissertation addresses various real-world problems that uncertainty creates for off-road mobile robots. A key theme of this work is to reconsider the contract between perception and planning to better tackle uncertainty. Simply passing a cropped map to planning without considering uncertainty or what planning needs from the perception handicaps the combined system. We have reconsidered this contract in three ways. First, we quantify uncertainty as a distribution over maps instead of relying on a single maximum-likelihood estimate. Second, we go beyond mapping by adjusting the target direction for planning. Third, we train both perception and planning modules from raw camera data, enabling fast adaptation. But these approaches are just scratching the surface of possible improvements to both perception and planning's contract and off-road autonomy in general.

This thesis was completed during the deployment heavy RACER program. Throughout the six experiment cycles, various robots had to navigate unseen deserts, forests, and scrublands at operational tempo. Robots and autonomy were pushed to the limit, with robots often experiencing catastrophic collisions on complex terrain. We had the opportunity to see what works and what ongoing challenges exist in off-road autonomy. This final chapter reviews the major takeaways from this experience.

Frequent Standardized Full-System Field Testing The major takeaways from PerceptOR and LAGR were frequent field testing on unseen environments and a standardized testing platform. RACER did both to great success. By providing fully equipped Racer Fleet and Racer Heavy platforms to researchers, RACER enabled immediate focus on autonomy development rather than sensor placement. Field testing at DirtFish [83] and later Ellensburg Washington enabled UW researchers to quickly identify the real problems. With robots at this size, logistics and field teams played a crucial role in keeping test tempo up and robots moving.

The keys to getting the most out of testing was full-system integration tests and unseen environments. With any new component or perception model, it proved vital to ensure it did not create downstream problems with other components of the stack and full-system tests verified that. For example, a perception model can have low validation loss but produces most errors at the edge of the costmap affecting mid-range planning more. While UW was limited to only a few test sites in Washington,

RACER tests enabled testing in diverse unseen locations. This was crucial to prevent overfitting the system to UW test sites. Further, experiencing new test sites enabled researchers to identify what are common problems across environments and what are unique to particular environments. Common problems could then be the focus of development as solving them had a larger impact.

The Mapping, Planning, and Control Paradigm This paradigm has been ubiquitous on off-road autonomy stacks since PerceptOR and still today [73; 107]. Learned perception models that build a birds eye view map provides a compact representation easy for both planning and human interpretation. While this thesis argues maps are insufficient to tackle the full problem, they remain an effective tool for local navigation. Further, the hierarchical planning pipeline which marries slower long horizon reasoning with fast short horizon local planning and control effectively enables full usage of the local map. This modular approach starkly differs from full end-to-end systems becoming popular in other fields of robotics [3; 10]. While it remains to be seen if end-to-end autonomy is possible, the modularity of this approach provides a list of key benefits. First, the separate component outputs provide interpretability for users enabling faster debugging and greater trust. Second, components provide redundancy increasing robustness. For example, local planning can plan around a surprise obstacle before mid-range planning has time to react. Third, modular components enable training from diverse sources. For example, an end-to-end policy must learn from sensor to action data obtained from a robot, but a perception model can learn to distinguish obstacles from just videos as shown in Chapter 5.

6.1 *Remaining Challenges & Future Work*

6.1.1 *Intercomponent communication – bottlenecks and downstream tasks*

With the modular autonomy system shown in Fig. 1.1 there exist two major challenges from which most of the development time was spent during RACER. First, by passing only a compact map to planning and a high-level plan to local control, we create information bottlenecks. For example, as discussed in Chapter 3, a singular most likely estimate map passed to planning removes any uncertainty information planning could use. Designing the information that is passed from module to the next can be challenging and may differ greatly from task to task. For example, a costmap is sufficient for general navigation but road following requires passing additional semantics to planning. Designing and implementing what information is passed can be tedious and difficult in practice.

Second, independent module training/development is agnostic of downstream uses. For example, perception may train a birds-eye-view classifier with a loss that weights all map cells equally [86], but downstream planning may prefer high accuracy mapping near the robot over further out. Another example is if planning development focuses on planning faster, but ignores uncertainty. Faster planning can actually increase plan switching from perception uncertainty, as it quickly adjusts to fluctuating costmap

obstacles, causing problems for control.

These challenges can be mitigated by frequent integration testing, but still require tedious manual tuning and adjustment. Recent work [121] has begun to consider modular end-to-end approaches that retain interpretability and robustness from modularity while also allowing the system to learn what information is relevant for each tasks. In essence, they leverage a shared sensor embedding space for use in mapping and planning and train both jointly. This co-training enables the embeddings to contain relevant information for both mapping and planning without having to design what information is relevant. They find co-training on multiple tasks performs significantly better than training on one task alone. Interestingly they also find additionally passing mappings outputs to planning have extra little benefit. Future work should consider similar designs for off-road autonomy.

6.1.2 Modularity and Monolithism

Monolithic architectures have gained prominence alongside large vision-language models [3; 10; 102]. These models have largely been applied to manipulation tasks. Their architectures feature a large, pre-trained Vision-Language Model (VLM) [11] backbone feeding embeddings to a smaller and faster diffusion policy [16] head. The model is trained on a large corpus of human teleoperated data to map images and language to low-level controls. The advantages of this approach are that a robot can leverage large-scale pre-training on non-robot data via the VLM and also utilize the language input of the VLM. However, they have a few disadvantages worth mentioning. Most notably, the model lacks interpretability as it is a black box with only the low-level controls as output. With no interpretability, it is hard to tell how well the model is performing without watching the output, which could sometimes be dangerous in off-road conditions. This issue is amplified by using behavior cloning to train these models, which is subject to compounding errors [78] as task horizons get longer. Finally, because it is monolithic, it can only be trained for robotics as a single unit (one large backpropagation), limiting its data sources to those that include low-level controls.

Today's spectrum spans hand-designed modular components at one end and monolithic approaches at the other. In the middle, benefiting from each design, are end-to-end modular systems. Approaches like PARA-Drive [121] provide a blueprint for modular end-to-end systems. Their system is entirely learned, easily enabling the use of pre-trained VLMs to create useful embeddings that combine images and language. And by being modular, they also provide interpretable intermediate outputs, and redundancy. A modular system for off-road autonomy could use shared embeddings (no bottlenecks) to feed multiple parallel heads: one for mapping, one for planning, and possibly one for LRN. By having multiple heads, we can also use multiple losses in addition to a behavior cloning loss. For example, if only video data is available, we could train the planning module (See Chapter 5) but not mapping, since we lack LiDAR. Training planning would also help adapt the shared embeddings to the new environment. While the exact architecture remains an open question, we argue combining modularity and end-to-end learning can capture the benefits of both approaches

to create robust off-road autonomy of the future.

6.1.3 *Fast Adaptation*

In off-road autonomy, there is a diverse set of environments the robot could operate in. Unlike on-road driving where most roads contain similar structure, navigating a forest is significantly different than driving over sand dunes. It is unlikely that we can expect strong zero-shot performance on the full diversity of environments the robot may experience. The focus then becomes on fast adaptation, how quickly can we retrain our robot to perform effectively in a new environment? Chapter 5 presents a novel way to quickly adapt without robot data. There is also recent work on online traversability learning [32; 92] for perception. While promising, this space is still underexplored. Little attention has been paid to how planning and control can adapt to new terrains. For example, a robot accustomed to hard-packed ground may struggle when tasked to climb a soft sand dune. Promising approaches to tackling this could be imitation learning from a human demonstrations with additional reinforcement learning finetuning [58]. Particularly with off-road vehicles like the Racer Fleet Vehicle, high quality human demonstrations are easy to collect. It may be difficult to train a policy to perform well in all environments, but isolated policies for particular challenges (e.g., ditches, sand dunes, etc), that are selectively used may be more tractable and should be considered for future work.

6.1.4 *Uncertainty Reasoning*

This thesis has covered tractable uncertainty reasoning in depth for planning. But for full system deployment some notable challenges still exist. First is quantifying uncertainty in perception. As noted in Chapter 3 simply having semantic class uncertainty per map cell is insufficient as it assumes cells are mutually exclusive. Future work should consider generative models like diffusion that enable sampling over the posterior of possible maps. Though diffusion has been used to extend the map horizon [28], full-map diffusion and real-world deployment still require further exploration. Another challenge with uncertainty is path switching, when the planner switches paths frequently between timesteps, confusing control. While DREAMS showed it can reduce path switching from uncertainty, it still remains a challenge in planning. Path switching is both a by-product of perception uncertainty and the choice of passing a singular path to local planning to follow. Possible other representations like cost-to-go maps [41], which can be learned or obtained from planning's full search tree, could be explored to allow local control agency on whether to switch direction.

6.2 *Closing Thoughts*

Off-road autonomy has come a long way since its beginnings in the late 1980s. We are now seeing robots navigate at high speeds in complex, unstructured environments with minimal human interventions. As robots transition from research projects to real-world applications, uncertainty reasoning will be more important than ever. This

dissertation is a small step in that direction. We hope it can jump-start future work that enables safe and reliable navigation for off-road robots.

Finally, throughout this dissertation, we have found that an effective approach was optimism in the face of uncertainty—not only for autonomous robots but also for life. While the fog of uncertainty is ever-present, it is those who navigate it with an open , positive mind who ultimately find what they need.

7
Bibliography

[1] U.S. Army CCDC Army Research Laboratory Public Affairs. Army research advances autonomous systems. Technical report, U.S. Army Combat Capabilities Development Command, Army Research Laboratory, 2019.

[2] Defense Advanced Research Projects Agency. Grand challenge 2004 final report. Technical report, Defense Advanced Research Projects Agency, 2004.

[3] Figure AI. Helix: A vision-language-action model for generalist humanoid control. Technical report, Figure AI, 2025.

[4] J. Asmuth, L. Li, M. L. Littman, A. Nouri, and D. Wingate. A bayesian sampling approach to exploration in reinforcement learning. *Conference on Uncertainty in Artificial Intelligence*, 2012.

[5] Inhwan Bae, Young-Jae Park, and Hae-Gon Jeon. Singulartrajectory: Universal trajectory predictor using diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[6] James Andrew Bagnell, David Bradley, David Silver, Boris Sofman, and Anthony Stentz. Learning for autonomous navigation. *IEEE Robotics & Automation Magazine*, 2010.

[7] M. Bajracharya, B. Tang, A. Howard, M. Turmon, and L. Matthies. Learning long-range terrain classification for autonomous navigation. In *IEEE International Conference on Robotics and Automation*, 2008.

[8] F. S. Barbosa, B. Lacerda, P. Duckworth, J. Tumova, and N. Hawes. Risk-aware motion planning in partially known environments. *Conference on Decision and Control*, 2021.

[9] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *Proceedings of the 2024 European Conference on Computer Vision (ECCV)*, 2024.

[10] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[11] Florian Bordes, Richard Yuanzhe Pang, Anurag Ajay, Alexander C. Li, Adrien Bardes, Suzanne Petryk, Oscar Mañas, Zhiqiu Lin, Anas Mahmoud, Bargav Jayaraman, Mark Ibrahim, Melissa Hall, Yunyang Xiong, Jonathan Lebensold, Candace Ross, Srihari Jayakumar, Chuan Guo, Diane Bouchacourt, Haider Al-Tahan, Karthik Padthe, Vasu Sharma, Hu Xu, Xiaoqing Ellen Tan, Megan Richards, Samuel Lavoie, Pietro Astolfi, Reyhane Askari Hemmat, Jun Chen, Kushal Tirumala, Rim Assouel, Mazda Moayeri, Arjang Talattof, Kamalika Chaudhuri, Zechun Liu, Xilun Chen, Quentin Garrido, Karen Ullrich, Aishwarya Agrawal, Kate Saenko, Asli Celikyilmaz, and Vikas Chandra. An introduction to vision-language modeling. *arXiv preprint arXiv:2405.17247*, 2024.

[12] Wolfram Burgard, Oliver Brock, and Cyrill Stachniss. Online learning for off-road robots: Using spatial label propagation to learn long-range traversability. In *Robotics: Science and Systems III*, 2008.

[13] X. Cai, M. Everett, J. Fink, and J. P. How. Risk-aware off-road navigation via a learned speed distribution map. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[14] Carlos Campos, Richard Elvira, Juan J. Gómez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map slam. *IEEE Transactions on Robotics*, 2021.

[15] M. Chen, E. Frazzoli, and D. Hsu. Pomdp-lite for robust robot planning under uncertainty. In *IEEE International Conference on Robotics and Automation*, 2016.

[16] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[17] J. J. Chung, A. J. Smith, R. Skeele, and G. A. Hollinger. Risk-aware graph search with dynamic edge cost discovery. *International Journal of Robotics Research*, 2019.

[18] Army Combat Capabilities Development Command. Scalable, adaptive, and resilient autonomy (sara) collaborative research alliance. Technical report, U.S. Army Research Laboratory, 2024.

[19] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. ISBN 0262033844.

[20] DARPA. Robotic autonomy in complex environments with resiliency (racer). https://www.darpa.mil/program/robotic-autonomy-in-complex-environments-with-resiliency, 2020.

[21] Defense Advanced Research Projects Agency. Grand challenge 2005 report to congress. Technical report, U.S. Department of Defense, 2005. URL https://www.grandchallenge.org/grandchallenge/docs/Grand_Challenge_2005_Report_to_Congress.pdf.

[22] D. Dey, A. Kolobov, R. Caruana, E. Kamar, E. Horvitz, and A. Kapoor. Gauss meets canadian traveler: Shortest-path problems with correlated natural dynamics. In *International Conference on Autonomous Agents and Multi-agent Systems*, 2014.

[23] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. TAP-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022.

[24] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. TAPIR: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10061–10072, 2023.

[25] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 1989.

[26] P. Eyerich, T. Keller, and M. Helmert. High-quality policies for the canadian traveler's problem. *AAAI Conference on Artificial Intelligence*, 2010.

[27] E Fahnestock. Guiding navigation of unknown environments with distant visual cues. Master's thesis, Massachusetts Institute of Technology, 2024.

[28] Ethan Fahnestock, Erick Fuentes, Philip R Osteen, Siddharth Ancha, and Nicholas Roy. Learning semantic traversability priors using diffusion models for uncertainty-aware global path planning. In *IEEE International Conference on Robotics and Automation*, 2024.

[29] D. Ferguson and A. Stentz. Field d*: An interpolation-based path planner and replanner. In *International Symposium on Robotics Research*, 2005.

[30] Seyedshams Feyzabadi and Stefano Carpin. Risk-aware path planning using hierarchical constrained markov decision processes. In *IEEE International Conference on Automation Science and Engineering*, 2014.

[31] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 1977.

[32] Jonas Frey, Matias Mattamala, Libera Piotr, Nived Chebrolu, Cesar Cadena, Georg Martius, Marco Hutter, and Maurice Fallon. Wild visual navigation: Fast traversability learning via pre-trained models and online self-supervision. In *Robotics: Science and Systems*, 2024.

[33] Yan Gao, Jing Wu, Xintong Yang, and Ze Ji. Efficient hierarchical reinforcement learning for mapless navigation with predictive neighbouring space scoring. *IEEE Transactions on Automation Science and Engineering*, 2023. DOI: 10.1109/TASE.2023.3312237.

[34] Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 1989.

[35] M. Guaman Castro, S. Triest, W. Wang, J. M. Gregory, F. Sanchez, J. G. Rogers III, and S. Scherer. How does it feel? self-supervised costmap learning for off-road vehicle traversability, 2023.

[36] Tyler Han, Alex Liu, Anqi Li, Alex Spitzer, Guanya Shi, and Byron Boots. Model predictive control for aggressive driving over uneven terrain. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[37] Benned Hedegaard, Ethan Fahnestock, Jacob Arkin, Ashwin Menon, and Thomas M. Howard. Discrete optimization of adaptive state lattices for iterative motion planning on unmanned ground vehicles. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[38] B. Hou and S. Srinivasa. Dynamic replanning with posterior sampling. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023.

[39] Thomas Howard. *Adaptive Model-predictive Motion Planning for Navigation in Complex Environments*. PhD thesis, Carnegie Mellon University, 2009.

[40] Wes Huang, Greg Grudic, and Larry Matthies. Editorial. *Journal of Field Robotics*, 2009.

[41] Jinwook Huh, Volkan Isler, and Daniel D. Lee. Cost-to-go function generating networks for high dimensional motion planning. In *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[42] D. Jackel, L. Douglas Hackett, Eric Krotkov, Michael Perschbacher, James Pippine, and Charles Sullivan. How darpa structures its robotics programs to improve locomotion and navigation. *Communications of the ACM*, 2007.

[43] L. 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*, 2006.

[44] Peng Jiang, Philip Osteen, Maggie Wigness, and Srikanth Saripalli. Rellis-3d dataset: Data, benchmarks and analysis. In *Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[45] Y. Jingkang, Z. Kaiyang, L. Yixuan, and L. Ziwei. Generalized out-of-distribution detection: A survey, 2022.

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

[47] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. *ECCV*, 2024.

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

[49] S. Koenig and M. Likhachev. D*lite. In *AAAI Conference on Artificial Intelligence*, 2002.

[50] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of Machine Learning Research*, 2021.

[51] Eric Krotkov, Scott Fish, Larry Jackel, Bill McBride, Mike Perschbacher, and Jim Pippine. The darpa perceptor evaluation experiments. *Autonomous Robots*, 2007.

[52] Ashish Kumar, Saurabh Gupta, and Jitendra Malik. Learning navigation subroutines from egocentric videos. *Proceedings in Machine Learning Research*, 2019.

[53] U.S. Army Research Laboratory. Robotics collaborative technology alliance (rcta). Technical report, U.S. Army Research Laboratory, 2018.

[54] Jacoby Larson, Mohan Manubhai Trivedi, and Michael H. Bruch. Off-road terrain traversability analysis and hazard avoidance for ugvs. In *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium*, 2011.

[55] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. *ara** : Anytime *a** with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, 2003.

[56] Z. W. Lim, D. Hsu, and W. S. Lee. Shortest path under uncertainty : Exploration versus exploitation. In *Conference on Uncertainty in Artificial Intelligence*, 2017.

[57] Qingze Tony Liu, Danrui Li, Samuel S. Sohn, Sejong Yoon, Mubbasir Kapadia, and Vladimir Pavlovic. Trajdiffuse: A conditional diffusion model for environment-aware trajectory prediction. In *Proceedings of the 27th International Conference on Pattern Recognition (ICPR)*, 2024.

[58] Yiren Lu, Justin Fu, George Tucker, Xinlei Pan, Eli Bronstein, Becca Roelofs, Benjamin Sapp, Brandyn White, Aleksandra Faust, Shimon Whiteson, Dragomir Anguelov, and Sergey Levine. Imitation is not enough: Robustifying imitation

with reinforcement learning for challenging driving scenarios. *arXiv preprint arXiv:2212.11419*, 2022.

[59] X. Meng, N. D. Ratliff, Y. Xiang, and D. Fox. Neural autonomous navigation with riemannian motion policy. *CoRR*, 2019.

[60] X. Meng, N. Hatch, A. Lambert, A. Li, N. Wagener, M. Schmittle, J. Lee, W. Yuan, Z. Chen, S. Deng, G. Okopal, D. Fox, B. Boots, and A. Shaban. Terrainnet: Visual modeling of complex terrain for high-speed, off-road navigation. In *Robotics: Science and Systems*, 2023.

[61] Takahiro Miki, Lorenz Wellhausen, Ruben Grandia, Fabian Jenelten, Timon Homberger, and Marco Hutter. Elevation mapping for locomotion and navigation using gpu. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2273–2280. IEEE, 2022.

[62] P. Moghadam, W. S. Wijesoma, and M. D. P. Moratuwage. Towards a fully-autonomous vision-based vehicle navigation system in outdoor environments. In *2010 11th International Conference on Control Automation Robotics & Vision*, 2010.

[63] P. Moghadam, S. Salehi, and W. S. Wijesoma. Computationally efficient navigation system for unmanned ground vehicles. In *2011 IEEE Conference on Technologies for Practical Robot Applications*, 2011.

[64] Riku Murai, Eric Dexheimer, and Andrew J. Davison. MASt3R-SLAM: Real-time dense SLAM with 3d reconstruction priors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2025.

[65] L. Murphy and P. Newman. Risky planning on probabilistic costmaps for path planning in outdoor environments. *IEEE Transactions on Robotics*, 2013.

[66] M. Ollis, W. H. Huang, M. Happold, and B. A. Stancil. Image-based path planning for outdoor mobile robots. In *IEEE International Conference on Robotics and Automation*, 2008.

[67] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[68] L. Orekhov, V. and H. Chung, T. The darpa subterranean challenge: A synopsis of the circuits stage. *Field Robotics*, 2022.

[69] M. W. Otte, S. G. Richardson, J. Mulligan, and G. Grudic. Local path planning in image space for autonomous robot navigation in unstructured environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[70] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Math. Oper. Res.*, 1987.

[71] C. H. Papadimitriou and M Yannakakis. Shortest paths without a map. In *Automata, Languages and Programming*, 1989.

[72] Archit Parnami and Minwoo Lee. Learning from few examples: A summary of approaches to few-shot learning. *ArXiv*, abs/2203.04291, 2022. URL https://api.semanticscholar.org/CorpusID:247318847.

[73] Manthan Patel, Jonas Frey, Deegan Atha, Patrick Spieler, Marco Hutter, and Shehryar Khattak. Roadrunner m&m – learning multi-range multi-resolution traversability maps for autonomous off-road navigation, 2024.

[74] Mikhail Pivtoraiko, Ross Alan Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 2009.

[75] Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 1989.

[76] William Qi, Ravi Teja Mullapudi, Saurabh Gupta, and Deva Ramanan. Learning to move with affordance maps. In *iclr*, volume abs/2001.02364, 2020.

[77] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[78] S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

[79] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for pomdps. *J. Artif. Int. Res.*, 2008.

[80] A. Sathyamoorthy, K. Weerakoon, T. Guan, J. Liang, and D. Manocha. Terrapn: Unstructured terrain navigation through online self-supervised learning, 2022.

[81] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018.

[82] R. Schmid, D. Atha, F. Schöller, S. Dey, S. Fakoorian, K. Otsu, B. Ridge, M. Bjelonic, L. Wellhausen, M. Hutter, and A. Agha-mohammadi. Self-supervised traversability prediction by learning to reconstruct safe terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[83] DirtFish Rally School. Home – dirtfish. Technical report, DirtFish Rally School, 2025.

[84] Pierre Sermanet, Marco Scoffier, Chris Crudele, Urs Muller, and Yann LeCun. Learning maneuver dictionaries for ground robot planning. In *Proceedings of the 2008 International Symposium on Robotics (ISR)*, 2008.

[85] Pierre Sermanet, Raia Hadsell, Marco Scoffier, Matt Grimes, Jan Ben, Ayse Erkan, Chris Crudele, Urs Miller, and Yann LeCun. A multirange architecture for collision-free off-road robot navigation. *Journal of Field Robotics*, 2009.

[86] A. Shaban, X. Meng, J. Lee, B. Boots, and D. Fox. Semantic terrain classification for off-road autonomous driving. In *IEEE International Conference on Robotics and Automation*, 2022.

[87] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine. ViNG: Learning Open-World Navigation with Visual Goals. In *IEEE International Conference on Robotics and Automation*, 2021.

[88] Yan Shen, Yu Li, and Zengping Li. Application of intelligent inspection robot in coal mine industrial heritage landscape: Taking wangshiwa coal mine as an example. *Frontiers in Neurorobotics*, 2022.

[89] D. Silver and J. Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems*, 2010.

[90] D. Silver, B. Sofman, N. Vandapel, J. A. Bagnell, and A. Stentz. Experimental analysis of overhead data processing to support long range navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[91] D. Silver, J. A. Bagnell, and A. Stentz. High performance outdoor navigation from overhead data using imitation learning. In *Robotics: Science and Systems*, 2009.

[92] Matthew Sivaprakasam, Samuel Triest, Cherie Ho, Shubhra Aich, Jeric Lew, Isaiah Adu, Wenshan Wang, and Sebastian Scherer. Salon: Self-supervised adaptive learning for off-road navigation. In *Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[93] E. L. Sofman, B. Ratliff, J. A. Bagnell, J. Cole, N. Vandapel, and A. Stentz. Improving robot navigation through self-supervised online learning. *Journal of Field Robotics: Special Issue on Machine Learning Based Robotics in Unstructured Environments*, 2006.

[94] A. Somani, N. Ye, D. Hsu, and W. S. Lee. Despot: Online pomdp planning with regularization. In *Advances in Neural Information Processing Systems*, 2013.

[95] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration. *OOD Workshop Conference on Robot Learning*, 2023.

[96] Gregory J. Stein, Christopher Bradley, and Nicholas Roy. Learning over subgoals for efficient navigation of structured, unknown environments. *Conference on Robot Learning*, 2018.

[97] A. Stentz. The d* algorithm for real-time planning of optimal traverses. Technical report, Carnegie Mellon University, Pittsburgh, PA, 1994.

[98] Anthony Stentz, John Bares, Thomas Pilarski, and David Stager. The crusher system for autonomous navigation. Technical report, National Robotics Engineering Center, CMU, 2007.

[99] M. J. A. Strens. A bayesian framework for reinforcement learning. In *International Conference on Machine Learning*, 2000.

[100] Z. Sunberg and M. J. Kochenderfer. Online algorithms for pomdps with continuous state, action, and observation spaces. In *International Conference on Automated Planning and Scheduling*, 2017.

[101] A. Suresh and S. Martínez. Planning under risk and uncertainty based on prospect-theoretic models. *arXiv preprint arXiv:1904.02851*, 2019.

[102] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, Konstantinos Bousmalis, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Oscar Chang, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, David D'Ambrosio, Sudeep Dasari, Todor Davchev, Coline Devin, Norman Di Palo, Tianli Ding, Adil Dostmohamed, Danny Driess, Yilun Du, Debidatta Dwibedi, Michael Elabd, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Leonard Hasenclever, Nicolas Heess, Brandon Hernaez, Alexander Herzog, R. Alex Hofer, Jan Humplik, Atil Iscen, Mithun George Jacob, Deepali Jain, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Chase Kew, Jerad Kirkland, Sean Kirmani, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Jacky Liang, Yixin Lin, Sharath Maddineni, Anirudha Majumdar, Assaf Hurwitz Michaely, Robert Moreno, Michael Neunert, Francesco Nori, Carolina Parada, Emilio Parisotto, Peter Pastor, Acorn Pooley, Kanishka Rao, Krista Reymann, Dorsa Sadigh, Stefano Saliceti, Pannag Sanketi, Pierre Sermanet, Dhruv Shah, Mohit Sharma, Kathryn Shea, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Jost Tobias Springenberg, Rachel Sterneck, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Vincent Vanhoucke, Jake Varley, Grace Vesom, Giulia Vezzani, Oriol Vinyals, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Yuxiang Yang, Rui Yao, Sergey Yaroshenko, Wenhao Yu, Wentao Yuan, Jingwei Zhang, Tingnan Zhang, Allan Zhou, and Yuxiang Zhou. Gemini

robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[103] Zachary Teed and Jia Deng. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *Advances in neural information processing systems*, 2021.

[104] Siyu Teng, Xuemin Hu, Peng Deng, Bai Li, Yuchen Li, Yunfeng Ai, Dongsheng Yang, Lingxi Li, Zhe Xuanyuan, Fenghua Zhu, and Long Chen. Motion planning for autonomous driving: The state of the art and future perspectives. *IEEE Transactions on Intelligent Vehicles*, 2023.

[105] W. R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

[106] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 2006.

[107] Marco Tranzatto, Mihir Dharmadhikari, Lukas Bernreiter, Marco Camurri, Shehryar Khattak, Frank Mascarich, Patrick Pfreundschuh, David Wisth, Samuel Zimmermann, Mihir Kulkarni, Victor Reijgwart, Benoit Casseau, Timon Homberger, Paolo De Petris, Lionel Ott, Wayne Tubby, Gabriel Waibel, Huan Nguyen, Cesar Cadena, Russell Buchanan, Lorenz Wellhausen, Nikhil Khedekar, Olov Andersson, Lintong Zhang, Takahiro Miki, Tung Dang, Matias Mattamala, Markus Montenegro, Konrad Meyer, Xiangyu Wu, Adrien Briod, Mark Mueller, Maurice Fallon, Roland Siegwart, Marco Hutter, and Kostas Alexis. Team cerberus wins the darpa subterranean challenge: Technical overview and lessons learned. *Field Robotics*, 2024. URL https://www.journalfieldrobotics.org/Field_Robotics/SI_DARPA_SubT_Final_files/Vol4_09.pdf.

[108] J. Tu, C. Liu, M. Wang, L. Gong, and Y. Li. Far-field terrain perception using max-margin markov networks. In *Intelligent Robotics and Applications*, 2012.

[109] Chris Urmson, Chris Baker, John Dolan, Paul Rybski, Bryan Salesky, William "Red" Whittaker, Dave Ferguson, and Michael Darms. Autonomous driving in traffic: Boss and the urban challenge. *AI Magazine*, 2009.

[110] S. Uryasev and R. T. Rockafellar. *Conditional Value-at-Risk: Optimization Approach*, pages 411–435. Springer US, 2001.

[111] U.S. Department of Transportation and White House Office of Science and Technology Policy. Ensuring american leadership in automated vehicle

technologies: Automated vehicles 4.0. Technical report, U.S. Government, 2020. URL https://www.transportation.gov/sites/dot.gov/files/2020-02/EnsuringAmericanLeadershipAVTech4.pdf.

[112] Lieve Van Woensel, Geoff Archer, Laura Panades-Estruch, and Darja Vrscaj. Ten technologies which could change our lives: Potential impacts and policy implications. Technical report, European Parliamentary Research Service, 2015. URL https://www.europarl.europa.eu/EPRS/EPRS_IDAN_527417_ten_trends_to_change_your_life.pdf.

[113] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. RoboTAP: Tracking arbitrary points for few-shot visual imitation. *International Conference on Robotics and Automation*, pages 5397–5403, 2024.

[114] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. In *IEEE International Conference on Robotics and Automation*, 2024.

[115] G. G. Waibel, T. Löw, M. Nass, D. Howard, T. Bandyopadhyay, and P. V. K. Borges. How rough is the path? terrain traversability estimation for local and global path planning. *Trans. Intell. Transport. Sys.*, 2022.

[116] M. Wang, J. Zhou, J. Tu, and C. Liu. Learning long-range terrain perception for autonomous mobile robots. *International Journal of Advanced Robotic Systems*, 2010.

[117] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2024.

[118] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter. Where should i walk? predicting terrain properties from images via self-supervised learning. *IEEE Robotics and Automation Letters*, 2019.

[119] Lorenz Wellhausen, René Ranftl, and Marco Hutter. Safe robot navigation via multi-modal anomaly detection. *IEEE Robotics and Automation Letters*, 5:1326–1333, 2020.

[120] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. In *Robotics: Science and Systems*, 2023.

[121] Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. Paradrive: Parallelized architecture for real-time autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[122] Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 2017.

[123] A. Wilson, A. Fern, S. Ray, and P. Tadepalli. Multi-task reinforcement learning: A hierarchical bayesian approach. In *International Conference on Machine Learning*, 2007.

[124] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997. DOI: 10.1109/CIRA.1997.613851.

[125] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024.

[126] S. Yoon, A. Fern, and R. Givan. Ff-replan: A baseline for probabilistic planning. In *International Conference on Automated Planning and Scheduling*, 2007.

[127] S. Yoon, A. Fern, R. Givan, and S. Kambhampati. Probabilistic planning via determinization in hindsight. In *AAAI Conference on Artificial Intelligence*, 2008.

[128] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.

[129] W. Zhang, Q. Chen, W. Zhang, and X. He. Long-range terrain perception using convolutional neural networks. *Neurocomputing*, 2018.

[130] Yunchu Zhang, Zhengyu Zhang, Liyiming Ke, Siddhartha Srinivasa, and Abhishek Gupta. ATK: Automatic task-driven keypoint selection for policy transfer from simulation to real world. In *Proceedings of the CoRL 2024 Workshop on Mastering Robot Manipulation in a World of Abundant Data*, 2024.