Learning Smooth State-Dependent Traversability from Dense Point Clouds

Zihao Dong[1], Alan Papalia[1,2], Leonard Jung[1], Alenna Spiro[1], Philip R. Osteen[3*], Christa S. Robison[3*], Michael Everett[1]
[1]Northeastern University [2]University of Michigan, Ann Arbor
[3]DEVCOM Army Research Laboratory (ARL) [*]Equal Contribution

Abstract: A key open challenge in off-road autonomy is that the traversability of terrain often depends on the vehicle's state. In particular, some obstacles are only traversable from some orientations. However, learning this interaction by encoding the angle of approach as a model input demands a large and diverse training dataset and is computationally inefficient during planning due to repeated model inference. To address these challenges, we present SPARTA, a method for estimating approach angle conditioned traversability from point clouds. Specifically, we impose geometric structure into our network by outputting a smooth analytical function over the 1-Sphere that predicts risk distribution for any angle of approach with minimal overhead and can be reused for subsequent queries. The function is composed of Fourier basis functions, which has important advantages for generalization due to their periodic nature and smoothness. We demonstrate SPARTA both in a high-fidelity simulation platform, where our model achieves a 91% success rate crossing a 40m boulder field (compared to 73% for the baseline), and on hardware, illustrating the generalization ability of the model to real-world settings. Our code will be available at https://github.com/neu-autonomy/SPARTA.

Keywords: Off-road Autonomy, Traversability, Geometric Deep Learning

1 Introduction

Many applications for autonomous mobile robots involve traversing rough, off-road environments [1, 2, 3]. One key to reliably navigate in these environments is to estimate terrain traversability, i.e., the ability to understand the interaction between the robot and the terrain, and assess relevant risk metrics (e.g., potential vehicle damage). Despite recent success in identifying traversable terrain [4, 5, 6, 7], motion planning in geometrically complex environments (e.g., dense forests, boulder fields) remains a challenge. Specifically, the risk associated with traversing a given terrain is often a complicated function of the robot's state, for example its speed and angle of approach, and the terrain geometry[1]. It is especially unclear how to account for the angle of approach (e.g., using hand-crafted risk heuristics based on terrain elevation [8, 9, 10]), as this requires both detailed terrain information and nuanced consideration of the underlying geometry.

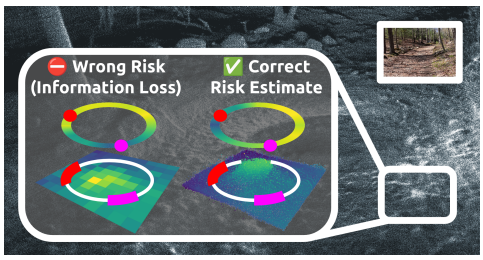


Figure 1. Dense point cloud captures geometric detail of the environment (Top right), while discretizing it into elevation map results in information loss. The underlying terrain is clearly distinguishable from noise (leaves, sensor noise) in the point cloud, but doing so is hard in an elevation map, leading to wrong risk estimation (Viridis Circle where Green denotes low risk).

In contrast, learning-based approaches have promise in being able to flexibly model the complex relationship between terrain and traversal risk. In particular, point cloud-based approaches are suit-

[1]We focus on angle-dependency in this paper, and leave the extension to other state variables to future work.

able for the problem setting, because point clouds are capable of capturing fine geometric details of the environment that would be missed in other representations like an elevation map. As shown in Fig. 1, in the point cloud, the terrain (ground) is distinguishable from noise (e.g. leaves). As a result, a point cloud-based method would prefer the safe approach angle (Red) over the risky one (Magenta) due to a relatively smooth elevation change. However, noise and terrain cannot be easily separated in the elevation map despite its high resolution (5cm). The terrain appears smoother from the risky angle (Magenta) thus an elevation map-based method would (incorrectly) select it instead.

In this paper, we propose a technique to learn risk variables of interest, based on a point cloud representation of the terrain. Specifically, to model system noise implicitly captured in the dataset, we represent risk as a categorical distribution due to its ability to approximate arbitrary distributions [4]. However, a point cloud-based network that naively encodes the angle of approach would be hard to train in practice due to the sparsity of data in robotics applications. As a result, our model takes in the point cloud and predicts an analytical function (represented using Fourier basis functions) that maps angle of approach to the target risk variable distribution. The choice of Fourier basis ensures that the resulting function is periodic, thus avoiding the wrap-around discontinuity [11], and has important advantages for generalization in the training process due to its upper-bounded Lipschitz constant by construction. The function, once computed for a terrain patch, can be queried efficiently with minimal computation overhead for any angle of approach at any timestep when integrated with modern planners like MPPI [12, 13]. To summarize, in this paper we propose SPARTA (**S**mooth **P**oint-cloud **A**pproach-angle **R**easoning for **T**errain **A**ssessment) with the following key contributions:

- A novel learning-based approach leveraging Fourier basis functions, which estimates angle-of-approach dependent traversal risk for complex terrains represented as point clouds.
- Theoretical analysis of the advantage of imposing smoothness by using Fourier basis functions, with connection to Lipschitz continuity and model generalization capability.
- Evaluations of the proposed pipeline through learning to predict potential vehicle damage, a risk variable prior works handle using handcrafted rules, with extensive results both in a high-fidelity simulator [14] and on hardware, with integration into a MPPI planner [12]. In Appendix G we discuss **SPARTA**'s generalization to other problem settings.

2 Related Work

Traversability Estimation Traversability estimation aims to identify terrain suitable for robot navigation. Traversability estimates typically rely on either manually designed features [8, 10] or are derived from learned vehicle movement patterns [15, 4, 6, 7, 5]. Both [8] and [10] use local elevation measurements to estimate stepping difficulty using heuristics such as step-wise elevation change. On the other hand, learned methods usually rely on multiple sensing modalities, especially elevation maps and semantic maps [15, 4, 6, 7, 5]. WVN [5] uses camera input and trains a traversable terrain classifier online leveraging pretrained features from DINO-ViT [16]. Meanwhile, [6, 7] fuse semantic and elevation measurements, and use separate decoder heads to predict dense semantic and elevation maps, which are used to score planner rollouts using various heuristics. [15, 4] use semantic and elevation maps to predict traversability distributions represented as linear/angular traction and incorporate the worst-case traction into the vehicle dynamics model for risk-aware planning. Several works modeled the state-dependency of traversability by encoding robot state as part of the neural networks' input [17, 18, 19]. However, there performances suffer from data sparsity and the dimensionality mismatch between exteroceptive inputs and low dimensional state space.

Learning Harmonic Functions Geometric deep learning has found its place in numerous machine learning applications. For example, FBM [20] uses Fourier basis functions to composite long-time series data and achieved state-of-the-art performance on multiple benchmarks. NeuRBF [21] replaces traditional grid-based structures with radial basis functions for constructing implicit neural representation of scenes, allowing the model to better adapt to the underlying signal structures. In robotics, OrbitGrasp [22] leverages spherical harmonics functions to represent a grasping quality

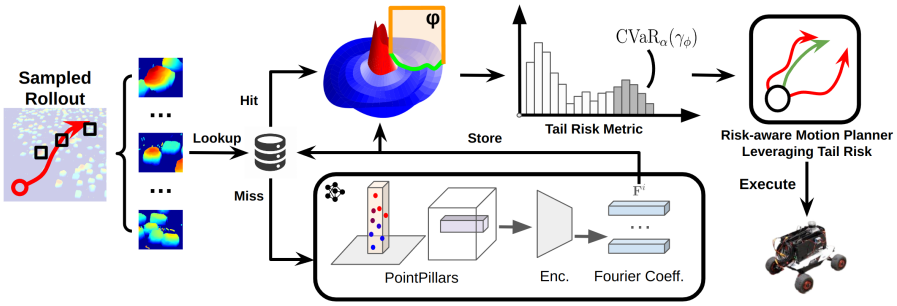


Figure 2. Overall pipeline of the proposed algorithm. For all terrain patches we are interested in, if its Fourier coefficients are not in the database, we compute them and store in the database. Otherwise we retrieve the Fourier coefficients from the database and construct the analytical functions, which are queried to compute the risk variable distribution, whose tail distribution is considered for risk aware planning.

function. Off-road autonomy, especially traversability estimation, is a setting with rich geometric structure, and as a result we propose to leverage Fourier basis functions to exploit this intrinsic prior.

Placement of This Work Unlike previous traversability estimation algorithms, SPARTA explicitly models the angle dependency of risk variables by exploiting the geometric structure of angle of approach (the 1-Sphere). Because we impose few assumptions on the risk variable of interest (i.e., it has clear lower and upper bounds and does not change abruptly with a small change in angle of approach), our approach could be used to extend (most) existing traversability estimation methods for more accurate and granular prediction. Besides tire deformation (demonstrated in our experiments Section 4), we show examples of variables with a similar geometric structure in Appendix G.

3 Method

In this section we introduce SPARTA, a method for predicting the risk associated with traversing a terrain patch at a given angle based on prior data traversing a wide variety of terrains (represented as dense point cloud). One challenge in traversability estimation is handling aleatoric uncertainty, the inherent uncertainty in a system which cannot be reduced with more data [23]. SPARTA captures this uncertainty by treating risk as a random variable – a categorical distribution with B discrete bins – whose concentration parameters are predicted by our model (Section 3.1 and Section 3.2). The tail of the distribution is used to estimate risk for risk-aware planning (Section 3.4).

The primary novelty in SPARTA is in representing this risk distribution. SPARTA predicts an analytical function for each bin, and each function smoothly maps any angle of approach to the corresponding risk concentration parameter (Section 3.2). We characterize the smoothness of these risk functions (Section 3.3), which we hypothesize to be useful for generalization across approach angles (empirically supported in Section 4.2). Once computed, the function can be re-used for subsequent queries for the same terrain patch, thus avoiding repeated neural network inference during planning. We include an experiment demonstrating the query efficiency of our model in Appendix C. An overview of the risk estimation pipeline and its application to planning is shown in Fig. 2.

3.1 Modeling Aleatoric Uncertainty with Categorical Distributions

In this work, we are interested in predicting a risk variable conditioned on the angle of approach. However, the aleatoric uncertainty inherent to the training data will likely prevent us from reliably estimating risk as a single scalar value. To develop robust real-world systems, it is important to properly account for aleatoric uncertainty [15, 4]. SPARTA captures aleatoric uncertainty by modeling risk as a random variable whose distribution reflects this uncertainty. Without assuming a specific parametric form, we approximate this distribution with a categorical distribution [4].

More formally, given a terrain patch represented as a point cloud $\mathbf{Q} \in \mathbb{R}^{m\times 3}$ where $m > 0$ is the number of points in the point cloud and the angle of approach $\phi \in [0, 2\pi]$, we are interested in obtaining a function $\Gamma\colon (\mathbf{Q}, \phi) \mapsto \mathbf{p}(\gamma_{\mathbf{Q},\phi})$, where $\gamma_{\mathbf{Q},\phi} \in \mathbb{R}$ denotes a risk related variable[2] and

[2]For example, risk related variables may include vehicle damage, terrain traction, or rollover risk.

$\mathbf{p}(\gamma_{\mathbf{Q},\phi})$ is the probability mass function (PMF) of a categorical distribution with B bins.[3] The categorical distribution $\mathbf{p}(\gamma_\phi)$ is parameterized by non-negative concentration parameters $\bar{\boldsymbol{\gamma}}_{\boldsymbol{\phi}} = [\gamma_\phi^1, \ldots, \gamma_\phi^B] \in \mathbb{R}^B_{\geq 0}$ and computed by normalizing $\bar{\gamma}_\phi$ to sum to 1:

$$\mathbf{p}(\gamma_\phi) = \frac{\bar{\boldsymbol{\gamma}}_{\boldsymbol{\phi}}}{\sum_{b=1}^{B} \gamma_\phi^b}. \tag{1}$$

To obtain this risk distribution $\mathbf{p}(\gamma_\phi)$, we can train a neural network on a dataset of traversals on various terrains, $\{(\mathbf{Q}_k, \mathbf{y}_k, \phi_k)\}_{k=1}^K$, where $\mathbf{y}_k$ denotes the ground truth (empirical) risk PMF. The network is trained to minimize the squared Earth Mover's Distance (EMD^2):

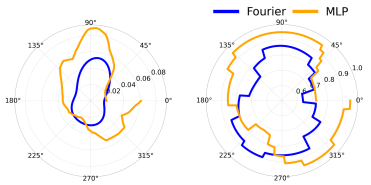
$$L^{\text{EMD}^2}(\mathbf{p}(\phi_k), \mathbf{y}_k) = ||\text{cs}(\mathbf{p}(\phi_k)) - \text{cs}(\mathbf{y}_k)||_2^2, \tag{2}$$

where $\text{cs}(\cdot)$ is the cumulative sum operator. However, training such a network for risk-aware planning poses two major difficulties. The first issue is due to the structure of angles of approach ϕ, i.e., there is a periodicity of 2π and we expect that small changes in ϕ should lead to small changes in the risk distribution. While a neural network can approximate this structure, it typically requires a large and diverse dataset to do so. The second issue is the model's query efficiency, which becomes a major concern for real-time planning. In subsequent sections, we discuss how SPARTA addresses these issues by leveraging Fourier basis functions to construct a smooth and periodic analytical function.

3.2 Leveraging S^1 as a Continuous Representation of Approach Angle

Here we introduce how SPARTA represents the risk concentration parameters γ_ϕ as a function of the angle of approach ϕ via Fourier basis functions. This representation stems from two main insights: (i) the set of angles $[0, 2\pi]$ can be mapped to the 1-sphere S^1 (i.e., the unit circle that lies in the terrain ground plane and is centered at the intersection of the terrain normal vector and the plane) and (ii) Fourier basis function is a fundamental way of representing arbitrary functions over S^1.

Given a representation of angles on S^1, we can consider the risk concentration parameters γ_ϕ as functions over S^1. It follows from harmonic analysis and representation theory [24] that any square-integrable real-valued function over the 1-sphere S^1 can be uniquely decomposed into a (possibly infinite) linear combination of the Fourier basis functions $\{1\} \cup \{\cos(k\phi), \sin(k\phi) : k \in [1, n]\}$. This suggests these basis functions as the natural choice for representing arbitrary functions on the 1-sphere. Altogether this allows us to compute risk concentration parameters at angle ϕ by taking a linear combination of the Fourier basis functions and applying the sigmoid function[4] $\sigma(\cdot)$ to ensure non-negativity (so the result is a valid concentration parameter):



(a) (b)

Figure 3. (a): Naive model (orange) suffers from wrap-around discontinuity, whereas our model (blue) predicts continuous signal. (b): The phenomenon propagates to the CVaR of the predicted distribution.

$$\gamma_\phi^i = \sigma\left(\frac{a_i^0}{2} + \sum_{k=1}^{n} [a_i^k \cos(k\phi) + b_i^k \sin(k\phi)]\right). \tag{3}$$

Concretely, our neural network Γ_θ takes an input point cloud $\mathbf{Q}$ and, for each bin $i \in [1, B]$, predicts the corresponding coefficients $\mathbf{F}^i \triangleq [a_i^0, a_i^1, b_i^1, \cdots a_i^n, b_i^n] \in \mathbb{R}^{2n+1}$. Eq. (3) is then used to compute the concentration parameters $\bar{\boldsymbol{\gamma}}_{\boldsymbol{\phi}}$ during planning. Eq. (3) is fast to compute as it requires a single (batched) dot product and a pass through the sigmoid function, and its runtime scales linearly with the maximum frequency n, a small number in practice[5]. We note that the choice of Fourier basis function naturally avoids wrap-around discontinuities at $\phi = 2\pi$. We demonstrate this in Fig. 3a, where we compare the predictions of a baseline multilayer perceptron model (orange) that encodes the angle of approach ϕ directly as a model input, and our model (blue). It can be seen that the

[3] For conciseness, we omit the subscript $\mathbf{Q}$ in the rest of this paper.

[4] While we used sigmoid function, other non-negative functions could theoretically be used (e.g., softplus).

[5] we found $n = 3$ to suffice in our experiments. We include model runtime comparison in Appendix C.

baseline model (orange) is discontinuous at the wrap-around point, resulting in a large jump in the predicted concentration parameters. This discontinuity is known to be harmful to model training when working with angle-related variables [11]. In contrast, our model (blue) naturally respects the periodicity and generates a globally smooth signal over S^1. Besides avoiding the wrap-around discontinuity, the Fourier basis functions allow better generalization when trained with limited data, as we detail through analyzing its Lipschitz constant in the next subsection.

3.3 Imposed Smoothness Improves Prediction Generalization

In off-road settings we may expect that small changes in approach angle ϕ induce relatively small changes in the risk variable γ_ϕ. In this subsection we show that γ_ϕ is smooth with respect to ϕ and provide a simple upper bound on the first derivative (i.e., an upper bound on the Lipschitz constant). We connect this Lipschitz constant analysis to results in learning theory [25, 26] to motivate the hypothesis that our representation improves generalization (see Section 4.2 for empirical evidence).

Formally, if L^i denotes the Lipschitz constant of the mapping from approach angle to risk concentration parameter $\phi \to \gamma^i_\phi$, then adding a small perturbation δ to ϕ bounds the change in γ^i_ϕ: $|\gamma^i_{\phi+\delta} - \gamma^i_\phi| \leq L^i|\delta|$. Our representation of the concentration parameters γ^i_ϕ guarantees that each γ^i_ϕ is smooth with respect to ϕ, as each γ^i_ϕ is a linear combination of sines and cosines. Furthermore, the Fourier coefficients $\mathbf{F}^i$ provide an upper bound on the Lipschitz constant L^i.

Theorem 1. *The concentration parameter γ^i_ϕ defined by the Fourier series representation in Eq. (3) is Lipschitz continuous with respect to ϕ with an upper bound on the Lipschitz constant L^i given by:*

$$L^i \leq \frac{1}{4}\sum_{k=1}^{n} k\sqrt{(a^k_i)^2 + (b^k_i)^2}. \tag{4}$$

Proof. See Appendix A. □

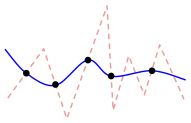


Figure 4. Demonstration of model in overfitting (red) and smooth-interpolation regime (blue).

Having a bounded Lipschitz constant is desirable, as small Lipschitz constants have been connected to improved generalization to unseen data in deep learning [27, 25, 26]. With this in mind, we consider Theorem 1 to suggest that our representation improves the model's ability to generalize to unseen angles of approach ϕ. As SPARTA attempts to estimate risk based on both point cloud data and the angle of approach $(\mathbf{Q}, \phi)$, we argue that an improved generalization over ϕ allows the learning process to focus on extracting meaningful information from the point cloud $\mathbf{Q}$, rather than having to simultaneously learn the relevant structure of the angle of approach ϕ. In other words, we believe that the smoothness and bounded Lipschitz constants represent informative geometric priors that allows the model to more efficiently learn relevant features. We provide empirical evidence for this in Section 4.2.

3.4 Incorporating the Learned Risk Model into a Motion Planner

Let $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^s$ denote the state of the robot, $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^u$ the control input, and subscript $t \geq 0$ the timestep. We assume the robot's motion evolves as a discrete time system, $\mathbf{x}_{t+1} = F(\mathbf{x}_t, \mathbf{u}_t)$. The planner optimizes for a control input sequence $\mathbf{u}_{0:T}$ given an initial state $\mathbf{x}_0$ that minimizes the worst-case planning objective. Specifically, we break the cost function into two parts: A task-related term $C(\mathbf{x}_{0:T})$ (e.g., goal reaching cost) and the worst-case risk computed from the risk distribution $\mathbf{p}(\gamma_{\phi_t})$. We adopt the Conditional Value at Risk (CVaR) as the worst-case risk because it is coherent and thus suitable for trustworthy risk assessment in robotics [28]. Specifically, the CVaR at level $\alpha \in [0, 1]$ of the risk variable γ_ϕ is $\text{CVaR}_\alpha(\gamma_\phi) := \frac{1}{\alpha}\int_0^\alpha \min\{\gamma \mid p(\gamma_\phi > \gamma) \leq \tau\} d\tau$, where the probabilities can be estimated from $\mathbf{p}(\gamma_\phi)$. Intuitively, $\text{CVaR}_\alpha(\gamma_\phi)$ represents the expectation of the last (right tail) α portion of the distribution $\mathbf{p}(\gamma_\phi)$. Notice that while the categorical distribution is discrete, so its CVaR is not necessarily continuous w.r.t. ϕ, our formulation guarantees the CVaR

does not suffer from a wrap-around discontinuity, as shown in Fig. 3b. To summarize, we adopt MPPI [12, 13] with the following planning problem:

$$
\begin{aligned}
\min_{\mathbf{u}_{0:T}} \quad & C(\mathbf{x}_{0:T}) + \sum_{t=0}^{T} \text{CVaR}_{\alpha}(\gamma_{\phi_t}) \\
\text{s.t.} \quad & \mathbf{x}_{t+1} = F(\mathbf{x}_t, \mathbf{u}_t) && \text{(Dynamics)} \\
& \mathbf{F}^i_{t+1} = \Gamma_\theta(\mathbf{Q}_{t+1})[i],\ \forall\, i \in [1, B] && \text{(Fourier Coefficients)} \\
& \gamma^i_{\phi_{t+1}} \leftarrow Eq.\ (3),\ \forall\, i \in [1, B] && \text{(Risk Concentration Params.)} \\
& \mathbf{p}_{t+1}(\gamma_{\phi_{t+1}}) \leftarrow Eq.\ (1) && \text{(Risk PMF)}
\end{aligned}
\tag{5}
$$

4 Experiments

In this section, we demonstrate the application of the proposed framework to assess potential vehicle damage (e.g., flat tires, broken axle). Although vehicle damage frequently occurs during real off-road autonomous operations, previous research has largely overlooked this risk metric or relied on relatively simple handcrafted heuristics (such as stepwise elevation change) to model it [8, 9, 10].

4.1 Tire Deformation As A Surrogate Metric

In the following experiments, we use tire deformation extent as a surrogate metric for the probability of vehicle damage. While our formulation can handle other risk-related variables (Appendix G) like wheel force and acceleration, which are also related to terrain contact, we use tire deformation because the tire is the part that directly contacts the terrain and thus is less susceptible to noise accumulated from simulating other vehicle parts. We collect training data using a high-fidelity simulation platform BeamNG.tech[6] [14]. Specifically, for a tire node w, let r_w denotes the distance from the (deformed) tire node to wheel center, r_{inner} and r_{outer} the radius of the wheel frame and non-deformed tire, respectively, and we compute the deformation extent as $d_w = \frac{r_w - r_{\text{inner}}}{r_{\text{outer}} - r_{\text{inner}}}$. Appendix D includes further details of the data collection process.

4.2 Model Performance

To demonstrate the generalization capability of our method, we compare its test set performance against a model that encodes the angle of approach as an input (*AngleInput*), where our model outperforms *AngleInput*. Our model and the *AngleInput* share the same architecture and have approximately the same number of trainable parameters, except that the *AngleInput* has an additional encoder head for the angle of approach, which is fused downstream with the point cloud feature to predict the concentration parameters. Details of the model architecture can be found in Appendix B. In Fig. 5, we show the test EMD^2 loss achieved by our model and the *AngleInput* with varying model sizes. The *AngleInput* baselines (Green and Orange) quickly overfit to the training data, with a best test loss of 0.15. In contrast, our test loss (Red and Blue) continues to decrease and eventually converged to 0.09. This is aligned with our hypothesis in Section 3.3, that our model would generalize better compared to *AngleInput* because the Fourier basis functions promise an upper-bound for the Lipschitz constant of the mapping $\phi \to \gamma^i_\phi$, whereas the *AngleInput* has to learn this mapping implicitly.

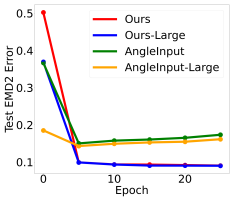


Figure 5. Our Model (Red and Blue) achieve lower test EMD^2 loss compared to *AngleInput* (Green and Orange) without overfitting.

4.3 Planning in Simulation

To test integration with a planner, we designed a challenging task where the vehicle needs to cross a randomly generated boulder field, with 20m radius and 1500 randomly placed obstacles, from 100 uniformly placed starting positions. Fig. 6 shows the overview of the environment.

[6] We choose BeamNG over conventional simulators for its high-fidelity soft body physics simulation.

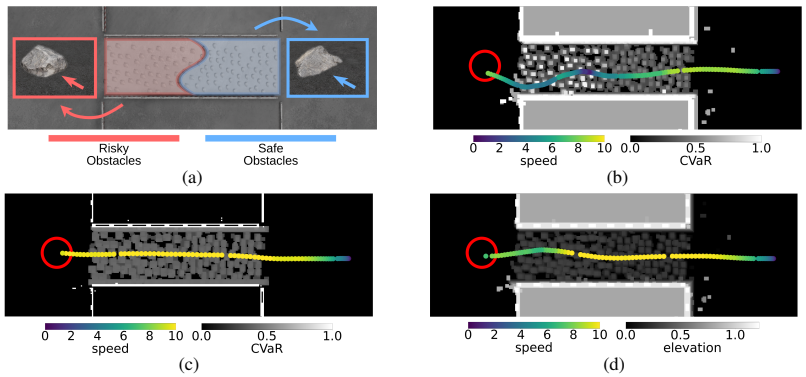


Figure 7. (a): Overview of the environment. The left half of the environment is filled with risky obstacles (Red), and the right half is filled with safe ones (Blue). Arrow indicates expected angle of approach. (b): Background denotes CVaR predicted by our method at the expected angle of approach. The planner is able to identify risky obstacles and decelerate preemptively. **Max Tire Force: 1426 N**. (c): Background denotes CVaR predicted by the *AngleFree*. It can not identify risky obstacles and the vehicle drives at a constant speed. **Max Tire Force: 2937 N**. (d): Background denotes elevation. Elevation baseline treat them as equally risky and traverse with a constant speed. **Max Tire Force: 4367 N**.

The obstacles are placed so densely that finding a clear path is challenging. Therefore, to complete the task at high speed, the planner must accurately distinguish between obstacles that can be safely traversed and those that pose significant risks. In Appendix E we provide an open loop experiment with ablation study to further demonstrate our approach. Besides *AngleInput*, we consider 2 baselines: (i) a model trained to predict the distributions without modeling the angle dependency (*AngleFree*), and (ii) a heuristic baseline that chooses the point with lowest maximum elevation under vehicle's wheel footprint (*Elev*). Since the obstacles are placed on flat ground, *Elev* is equivalent to evaluating stepping difficulty as in [8, 10]. We use a MPPI [12] planner with Ackermann kinematics, and the planning objective includes the distance to the goal, a velocity cost for driving over speed limit $\bar{v}$, and a risk cost:

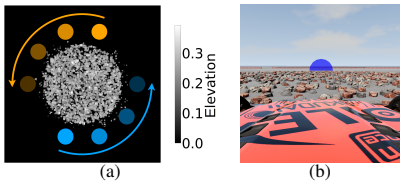


Figure 6. (a): Overview of the Boulder Field (40m) test environment with goals (blue) and starting points (orange), overlaid with an elevation map (grayscale). (b): Overview of the environment from driver's POV.

$$C_{\text{risk}}(t) = \begin{cases} \text{CVaR}_\alpha(\gamma_{\phi_t}) & \text{for Learned Models} \\ \max(h_{\text{FL}}, h_{\text{FR}}, h_{\text{RL}}, h_{\text{RR}}) & \text{for Elevation Baseline} \end{cases} \tag{6}$$

$$Cost = \underbrace{w_{\text{goal}} \sum_{t=0}^{T} D(\mathbf{x}, \mathbf{x}_{\text{goal}})}_{\text{Goal Cost}} + \underbrace{w_{\text{v}} \sum_{t=1}^{T} \mathbb{1}(v_t > \bar{v})}_{\text{Velocity Cost}} + \underbrace{w_{\text{risk}} \sum_{t=0}^{T} v_t C_{\text{risk}}(t)}_{\text{Risk Cost}} . \tag{7}$$

where $w_{\text{goal}}, w_{\text{v}}, w_{\text{risk}}$ are weights for associated cost term. Notice we scale the Risk Cost term by the current vehicle velocity v_t to penalize traversing risky terrain at high speed.

In Table 1 we report the success rate (S.R., reached goal without damage) and average vehicle velocity (Vel) for the methods tested. Our model achieves higher success rate compared to *AngleInput*, presumably because of its stronger generalization ability. Most importantly, the performances of models that captures angle-dependency (ours and *AngleInput*) are better than *AngleFree* and *Elev* because their Risk Cost term cannot distinguish between safe obstacles from risky ones. As a result, the vehicle attempts to traverse the boulder field at a roughly constant velocity, i.e. the Risk Cost in these cases degenerates to velocity cost. To test if the faster speed of *AngleFree* and *Elev* caused their low success rates, we tuned the weights and decreased $\bar{v}$ from 4 to 2 to enforce a slower velocity. However, their success rates do not significantly increase even traversing at a lower velocity, which suggests that they are ineffective risk estimators in this environment.

Algorithm	$\bar{v}$	S.R. (%) ↑	Vel (m/s) ↑
Ours	4	**91**	<u>2.6</u>
AngleInput	4	<u>85</u>	2.5
AngleFree	4	73	**3.2**
	2	79	1.9
Elev	4	73	**3.2**
	2	75	2.1

Table 1. Success rate (S.R.) of the models in the boulder field test. Our method achieves best success rate, and overall angle-dependent models perform better than those that do not consider angle. **Best**, and <u>second best</u>.

To further help understand how *AngleFree* and *Elev* underestimate traversal risk, we crafted an environment, where the vehicle has to go through a narrow passage way filled with obstacles. We

picked two obstacles with same elevation yet one appears to be safe and the other is visually risky, and use each of them fill half of the environment. Detail of the environment is shown in Fig. 7a, where red denotes the risky obstacles and blue denotes safe obstacles, when approached from the expected orientation (from right to left). The CVaR predicted by our model (Fig. 7b) correctly identifies the risky obstacles from the safe ones, and is manifested by the vehicle's preemptive deceleration. During the traversal, our model induced the lowest force on the tire 1426 N. The CVaR predicted by the *AngleFree* model (Fig. 7c) however, is very similar for all obstacles because the obstacles are all safe to traverse from certain orientations, and as a result the vehicle drives at a constant (high) velocity through the obstacles. Similar to *AngleFree*, the obstacles have same height and thus *Elev* drives the vehicle at a constant speed (until it hits an obstacle very hard and thus slowed down). Both *AngleFree* and *Elev* results in significantly higher tire force (2937 N and 4367 N respectively), signifying higher probability of vehicle damage. In other words, in order for *AngleFree* and *Elev* to safely drive the vehicle through complex obstacles, their maximum speed must be set conservatively such that the vehicle can safely traverse the most risky obstacle possible. However, empirically it is difficult to find such a threshold that guarantees safety.

4.4 Planning on Hardware

In this experiment, we demonstrate our model on a custom-built wheeled robot (specification can be found in Appendix F), and compare its performance qualitatively to *Elev*. We create an environment where a robot has to pick one out of 2 possible paths to reach its goal, where the left path is safe and the right path is risky. For the obstacle, we cut rigid foam into a shape resembling a traffic flow plate. We place the obstacle on the left path as shown in the cyan box in Fig. 8. To avoid vehicle damage and allow the robot to smoothly traverse, we *virtually* place the same obstacle rotated 180 degrees on the right path as the risky obstacle. The point clouds (pre-collected with [29]) are overlaid in Fig. 8 for demonstration purposes. The planner uses unicycle dynamics and the planning objective is the same as Eq. (7) except we do not include the velocity terms. The planner samples 500 rollouts with a horizon of 5.0s, and runs at 50 Hz with our model (70 Hz with *Elev*). In the white box in Fig. 8 we show the trajectories of our method (Green) and *Elev* (Orange). Our model correctly identifies the safe path for all 5 trials, yet *Elev* chooses the risky path for 3 out of 5 trials. While our model drives the vehicle safely and smoothly through the safe obstacles, *Elev* would have likely caused vehicle damage by ramming into sharp edges of an obstacle (red circle).

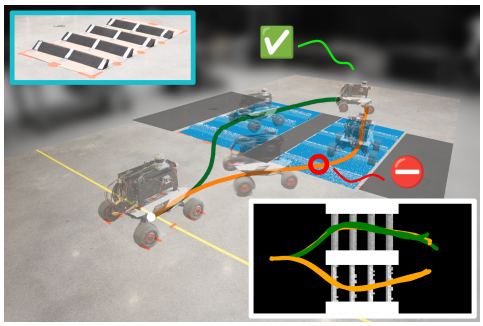


Figure 8. Environment overview with overlaid point cloud (blue, colored by elevation), physical obstacle (cyan box), and lethal obstacles (shaded). Our method (Green) consistently chooses the safe path, but *Elev* (Orange) can choose the risky path as the obstacles have same elevation change. Trajectories are shown in the white box and background denotes CVaR of the expected angle of traversal (Left to Right).

5 Conclusion

In this paper, we propose **SPARTA**, a general framework for estimating terrain traversal risk conditioned on vehicle angle of approach. Specifically, we identify and exploit the geometric structure in the problem setting, and propose to predict a smooth analytical function over the 1-Sphere to achieve better data efficiency and generalization performance. During planning, the function is queried to construct a categorical distribution for a risk variable of interest at any angle of approach, and the distribution's tail risk (CVaR) is considered to account for aleatoric uncertainty. Experiments in simulation and on hardware demonstrate the effectiveness of the proposed approach. Most importantly, off-road problems are naturally rich in geometric structure, and our formulation is general enough such that it finds application in numerous different problem settings. Future work may further explore the geometric structure in other off-road autonomy problems.

6 Limitation

Our work has several limitations. For example, in this work we assume the environment is fully observable, i.e. a prebuilt point cloud map is accessible during planning. While the point cloud map we can build on the fly is sufficiently dense, the model's performance without a prebuilt map is still to be tested. A potential solution to this problem is to penalize terrain with limited observation during planning or incorporate depth inpainting as in [7, 6]. In addition, this work focused on estimating traversability given geometric information, whereas semantic information is also important to consider in many off-road settings. We anticipate that future work could incorporate semantic information by performing sensor fusion with point pillar feature in birds-eye-view [7]. While the smoothness assumption in Section 3.3 is reasonable, settings that potentially break this assumption might exist in the real world. Quantitative analysis to identify such failure modes can be a valuable addition to our framework. Last but not least, tire deformation is a metric that's only available in simulation. However, we are not interested in predicting tire deformation in the real world, but rather using it as a surrogate metric for terrain property (i.e. how risky is the terrain). Future work could explore other variables to quantify vehicle damage.

Acknowledgments

This research was sponsored by the DEVCOM Army Research Laboratory (ARL) under SARA CRA W911NF-24-2-0017. Distribution Statement A: Approved for public release; distribution is unlimited.

References

[1] L. F. Oliveira, A. P. Moreira, and M. F. Silva. Advances in agriculture robotics: A state-of-the-art review and challenges ahead. *Robotics*, 10(2):52, 2021.

[2] I. Lluvia, E. Lazkano, and A. Ansuategi. Active mapping and robot exploration: A survey. *Sensors*, 21(7):2445, 2021.

[3] G. Wang, W. Wang, P. Ding, Y. Liu, H. Wang, Z. Fan, H. Bai, Z. Hongbiao, and Z. Du. Development of a search and rescue robot system for the underground building environment. *Journal of field robotics*, 40(3):655–683, 2023.

[4] X. Cai, S. Ancha, L. Sharma, P. R. Osteen, B. Bucher, S. Phillips, J. Wang, M. Everett, N. Roy, and J. P. How. Evora: Deep evidential traversability learning for risk-aware off-road autonomy. *IEEE Transactions on Robotics*, 2024.

[5] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter. Fast traversability estimation for wild visual navigation. *arXiv preprint arXiv:2305.08510*, 2023.

[6] X. Meng, N. Hatch, A. Lambert, A. Li, N. Wagener, M. Schmittle, J. Lee, W. Yuan, Z. Chen, S. Deng, et al. Terrainnet: Visual modeling of complex terrain for high-speed, off-road navigation. *arXiv preprint arXiv:2303.15771*, 2023.

[7] J. Frey, M. Patel, D. Atha, J. Nubert, D. Fan, A. Agha, C. Padgett, P. Spieler, M. Hutter, and S. Khattak. Roadrunner-learning traversability estimation for autonomous off-road driving. *IEEE Transactions on Field Robotics*, 2024.

[8] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi. Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation. *arXiv preprint arXiv:2103.02828*, 2021.

[9] A. Datar, C. Pan, and X. Xiao. Learning to model and plan for wheeled mobility on vertically challenging terrain. *IEEE Robotics and Automation Letters*, 2024.

[10] D. Lee, I. M. A. Nahrendra, M. Oh, B. Yu, and H. Myung. Trg-planner: Traversal risk graph-based path planning in unstructured environments for safe and efficient navigation. *IEEE Robotics and Automation Letters*, 2025.

[11] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5745–5753, 2019.

[12] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[13] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[14] BeamNG GmbH. BeamNG.tech. URL `https://www.beamng.tech/`.

[15] X. Cai, M. Everett, J. Fink, and J. P. How. Risk-aware off-road navigation via a learned speed distribution map. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2931–2937. IEEE, 2022.

[16] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[17] J. Frey, D. Hoeller, S. Khattak, and M. Hutter. Locomotion policy guided traversability learning using volumetric representations of complex environments. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5722–5729. IEEE, 2022.

[18] K. Weerakoon, A. J. Sathyamoorthy, U. Patel, and D. Manocha. Terp: Reliable planning in uneven outdoor environments using deep reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9447–9453. IEEE, 2022.

[19] J. Seo, T. Kim, K. Kwak, J. Min, and I. Shim. Scate: A scalable framework for self-supervised traversability estimation in unstructured environments. *IEEE Robotics and Automation Letters*, 8(2):888–895, 2023.

[20] R. Yang, L. Cao, J. YANG, et al. Rethinking fourier transform from a basis functions perspective for long-term time series forecasting. *Advances in Neural Information Processing Systems*, 37:8515–8540, 2024.

[21] Z. Chen, Z. Li, L. Song, L. Chen, J. Yu, J. Yuan, and Y. Xu. Neurbf: A neural fields representation with adaptive radial basis functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4182–4194, 2023.

[22] B. Hu, X. Zhu, D. Wang, Z. Dong, H. Huang, C. Wang, R. Walters, and R. Platt. Orbitgrasp: $se(3)$-equivariant grasp learning. *arXiv preprint arXiv:2407.03531*, 2024.

[23] E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110(3):457–506, 2021.

[24] W. Rudin. *Fourier analysis on groups*. Courier Dover Publications, 2017.

[25] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in neural information processing systems*, 32, 2019.

[26] G. Khromov and S. P. Singh. Some fundamental aspects about lipschitz continuity of neural networks. *arXiv preprint arXiv:2302.10886*, 2023.

[27] M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116 (32):15849–15854, 2019.

[28] A. Majumdar and M. Pavone. How should a robot assess risk? towards an axiomatic theory of risk in robotics. In *Robotics Research: The 18th International Symposium ISRR*, pages 75–84. Springer, 2020.

[29] K. Chen, R. Nemiroff, and B. T. Lopez. Direct lidar-inertial odometry: Lightweight lio with continuous-time motion correction. In *2023 IEEE international conference on robotics and automation (ICRA)*, pages 3983–3989. IEEE, 2023.

[30] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

A Derivation of Lipschitz Constant

With a slight abuse of notation F, Eq. (3) can be written in the following way:

$$G(\phi) = \frac{a_i^0}{2} + \sum_{k=1}^{n} [a_i^k \cos(k\phi) + b_i^k \sin(k\phi)] \tag{8}$$

$$F(\phi) = \sigma(G(\phi)) \tag{9}$$

where $\sigma(\cdot)$ is the sigmoid function and a_i^0, a_i^k, and b_i^k are the Fourier coefficients predicted by our network.

We will derive an upper bound on the Lipschitz constant of $F(\phi)$, via the chain rule:

$$|F'(\phi)| = |\sigma'(G(\phi))| \cdot |G'(\phi)|. \tag{10}$$

First we differentiate $G(\phi)$ w.r.t. ϕ:

$$G'(\phi) = \sum_{k=1}^{n} k a_i^k \cos(k\phi) - k b_i^k \sin(k\phi) \tag{11}$$

$$= \sum_{k=1}^{n} k [a_i^k \cos(k\phi) - b_i^k \sin(k\phi)]. \tag{12}$$

For every $k \in [1, n]$, let:

$$R_k = \sqrt{(a_i^k)^2 + (b_i^k)^2} \tag{13}$$

$$\Delta_k = \arctan(\frac{-b_i^k}{a_i^k}), \tag{14}$$

Then Eq. (12) can be rewritten as a phase shift leveraging the cosine addition formula:

$$G'(\phi) = \sum_{k=1}^{n} k [R_k \cos(k\phi) \frac{a_i^k}{R_k} - R_k \sin(k\phi) \frac{b_i^k}{R_k}] \tag{15}$$

$$= \sum_{k=1}^{n} k [R_k \cos(k\phi) \cos(\Delta_k) + R_k \sin(k\phi) \sin(\Delta_k)] \tag{16}$$

$$= \sum_{k=1}^{n} k R_k \cos(k\phi - \Delta_k). \tag{17}$$

Notice that the maximum value of a $\cos(\cdot)$ is 1, as a result, Eq. (17) is upper bounded by:

$$G'(\phi) \leq \sum_{k=1}^{n} k R_k \tag{18}$$

$$= \sum_{k=1}^{n} k \sqrt{(a_i^k)^2 + (b_i^k)^2}. \tag{19}$$

Because the sigmoid function is Lipschitz with a Lipschitz constant of $\frac{1}{4}$, by the chain rule:

$$|F'(\phi)| = |\sigma'(G(\phi))| \cdot |G'(\phi)| \tag{20}$$

$$\leq \frac{1}{4} \sum_{k=1}^{n} k\sqrt{(a_i^k)^2 + (b_i^k)^2}. \tag{21}$$

As a result the Lipschitz Constant of $F(\phi)$ is bounded by:

$$L^i \leq \frac{1}{4} \sum_{k=1}^{n} k\sqrt{(a_i^k)^2 + (b_i^k)^2}. \qquad \square$$

Note that the peak of the derivative of $G(\phi)$ and the sigmoid function $\sigma(\cdot)$ may not coincide, and as a result we expect the Lipschitz constant achieved in practice to be smaller.

B Model Architecture

All the learned models described in this paper (Ours, *AngleInput*, and *AngleFree*) predict categorical distributions composed of $B = 8$ bins, parameterized by the concentration parameters $[\gamma_\phi^1, \ldots, \gamma_\phi^8]$.

All models share the same Point Pillars encoder architecture [30], which extracts a feature representation from the input point cloud $\mathbf{Q}$. Specifically, the point cloud $\mathbf{Q}$ is first uniformly scaled to fit within a cube defined by $x \in [-0.5, 0.5]$, $y \in [-0.5, 0.5]$, and $z \in [0, 0.5]$. This normalized point cloud is then discretized into uniformly spaced pillars in the x-y plane, each with a grid size of $0.1m$. Points within each pillar are augmented with x_c, y_c, z_c, x_p, and y_p where the c subscript denotes distance to the arithmetic mean of all points in the pillar and the p subscript denotes the offset from the pillar's x, y center [30]. Notice we exclude the reflectance feature, yielding an 8-dimensional augmented input representation for each lidar point. For each pillar, the encoder samples up to 32 points and generates a corresponding 32-dimensional feature vector. The resulting output from the Point Pillars encoder is a pseudo-image tensor with dimensions $10 \times 10 \times 32$.

Next, we detail the remaining components of the model architecture.

B.1 Neural Network Used in SPARTA

The pseudo-image from the Point Pillars encoder [30] is subsequently processed by a two-layer convolutional backbone, specified as follows:

- Input Channels: [32, 64]
- Output Channels: [64, 256]
- Kernel Size: [3, 3]
- Stride: [2, 2]
- Padding: [1, 1]

Each convolutional layer in the backbone employs batch normalization and ReLU activation. The resulting output tensor, of dimensions $3 \times 3 \times 256$, is passed through a max pooling operation to yield a compact 256-dimensional point cloud feature representation.

To represent the smooth analytical mapping $\phi \rightarrow \gamma_\phi^i$, we employ Fourier basis functions up to a maximum frequency of $n = 3$, specifically the basis set $\{1, \cos(\phi), \sin(\phi), \cos(2\phi), \sin(2\phi), \cos(3\phi), \sin(3\phi)\}$. The choice of n is justified in Appendix E with an ablation study. Consequently, for each bin $i \in [1, 8]$, the network predicts 7 Fourier coefficients. Thus, the SPARTA decoder consists of a multilayer perceptron (MLP) that transforms the 256-dimensional point cloud features into a 56-dimensional output vector $[\mathbf{F}^1, \ldots, \mathbf{F}^8]$. This MLP decoder contains one hidden layer with 512 neurons and utilizes layer normalization and ReLU activation functions.

B.2 AngleInput

The pseudo-image from the Point Pillars encoder [30] is subsequently processed by a three-layer convolutional backbone, specified as follows:

- Input Channels: [32, 64, 128]
- Output Channels: [64, 128, 256]
- Kernel Size: [3, 3, 3]
- Stride: [2, 2, 2]
- Padding: [1, 1, 0]

The encoder for angle of approach ϕ is a MLP that maps 1-dimensional input ϕ to a 32-dimensional embedding with one hidden layer with 16 neurons. The decoder merges the point cloud feature (256-dimensional) and the angle feature (32-dimensional), and outputs the 8 concentration parameters $[\gamma_\phi^1, \ldots, \gamma_\phi^8]$, with a hidden layer with 64 neurons. Both MLPs utilizes layer normalization and ReLU activation functions.

B.3 AngleFree

The *AngleFree* baseline network uses the same convolutional backbone as *AngleInput*. A decoder head transforms the 256-dimensional point cloud feature to the 8 concentration parameters $[\gamma_\phi^1, \ldots, \gamma_\phi^8]$, with two hidden layers with [128, 64] neurons.

C Inference Efficiency

To demonstrate the query efficiency of our model compared to *AngleInput*, we measure their respective inference runtimes. Specifically, for *AngleInput*, we measure the runtime of encoding the approach angle, decoding the concentration parameters, and computing the categorical distribution (Eq. (1)). For our model, we measure the runtime associated only with computing the concentration parameters (Eq. (3)) and the categorical distribution (Eq. (1)). Note other components (e.g. extracting feature vector from point cloud) of both models can be precomputed or reused. Considering that more complex point clouds require higher-dimensional point cloud features and deeper network architectures, we perform a stress test by progressively increasing the number of layers in both models, as well as the maximum frequency n in our model. Both models process 25,000 queries with average runtimes reported in Fig. 9. Our model demonstrates an order-of-magnitude faster runtime compared to *AngleInput*. Most importantly, the runtime of our model remains constant despite increased complexity, as it only involves a single batched, low-dimensional dot product and sigmoid computation [7]. In contrast, the runtime of *AngleInput* increases with complexity.

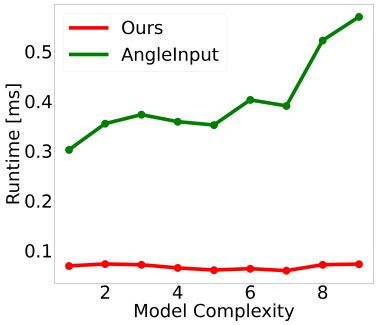


Figure 9. Runtime of *AngleInput* (Green) increases with model complexity, while the runtime of ours (Red) remains roughly constant. This is because during planning, we only need to perform a low dimensional dot product and a sigmoid pass during inference, whereas *AngleInput* requires repeated neural network inference. The x-axis denotes additional hidden layers in the decoder and additional frequencies in our model, i.e., $n \in [3, 13]$.

D Data Collection Detail

In this section we provide more details of our data collection process using the automotive simulator *BeamNG.tech* [14]. We intend to generate random obstacles for the vehicle to drive over, and collect

[7] The dimension of the dot product depends only on the choice of maximum frequency n. In fact the dimension is exactly $2n + 1$.

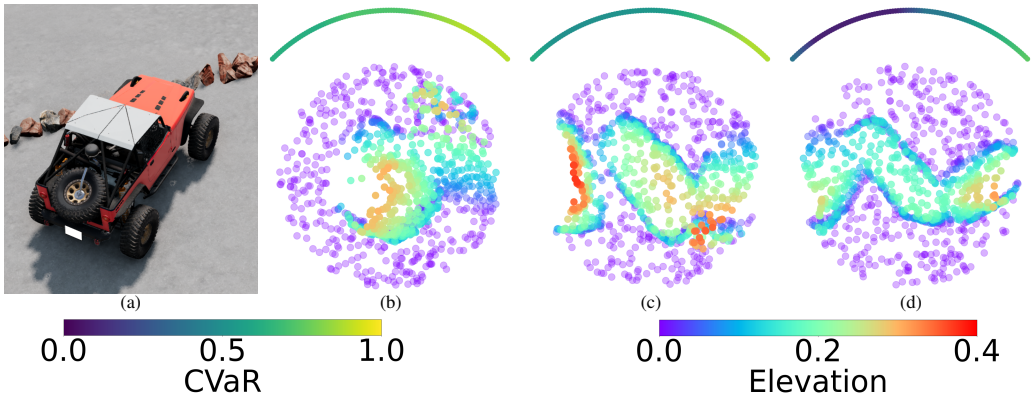


Figure 10. (a): Overview of the test environment and the vehicle. (b)-(d): Example point cloud (colored by elevation) and the CVaR (Viridis arces around the point clouds) predicted by our model at the candidate angles of approach.

the tire deformation during the traversal. We choose a patch size of $1 \times 1m$ that covers the footprint of the vehicle wheel, and randomly place (non-overlapping) obstacles in the patch. Maximum elevation of the obstacle is set to $0.5m$ (the vehicle's minimum ground clearance). For every patch, we collect its point cloud and downsample to a maximum of 1024 points. To minimize the influence of vehicle dynamics (for the purpose of generalization to unseen hardware), we place the vehicle such that only one wheel contacts the obstacle. The deformation extent $d_w = \frac{r_w - r_{\text{inner}}}{r_{\text{outer}} - r_{\text{inner}}}$ of all contacting tire node w are recorded during the traversal. We discard all samples below 0.2 (the tire deformation when the vehicle is static on flat ground) and divide the rest into a histogram with 8 bins, which forms the empirical distribution $\mathbf{y}$. In total, 24,000 samples $\{\mathbf{Q}_k, \mathbf{y}_k, \phi_k\}$ are collected. We augment the samples by applying 8 random rotations (to both the point cloud and the angle of approach), resulting in a total of 216,000 samples. We further add random perturbation to the point clouds for more robust performance. We use 90% of the data for training and 10% for testing.

E Traversing a Row of Obstacles with Ablation Study

In this experiment, we test our model's ability to identify safe traversal points and angles of approach in a row of packed, randomly placed obstacles. An overview of the environment, a few example point clouds (colored by elevation), and the CVaRs predicted by our model are shown in Fig. 10. This environment is simpler than the boulder field in Section 4.3 and thus helpful in isolating the influence of the variables we are interested in. We consider two baselines, the *AngleInput* and a heuristic baseline that chooses the point with lowest maximum elevation under vehicle's wheel footprint (*Elev*). Since the obstacles are placed on flat ground, the *Elev* baseline is equivalent to finding the plan with lowest stepping difficulty as in [8, 10]. As shown in Table 2, our model achieves higher success rate (S.R., the vehicle drives over the obstacle without damage) than the *AngleInput*, but both learning-based method significantly outperforms *Elev*. This is expected because the obstacles are spawned with similar sizes, and as a result *Elev* degenerates to random selection, leading to much lower success rate and thus high risk of vehicle damage.

Algorithm	α	n	Suc. ↑	Dmg. ↓
Ours	0.9	3	**95**	**5**
	0.9	1	91	9
	0.9	5	<u>94</u>	<u>6</u>
	0	3	89	11
AngleInput	0.9	–	92	8
	0	–	89	11
Elev	–	–	73	27

Table 2. Number of success (Suc.) and vehicle damage (dmg.) in simulation. Our model outperforms *AngleInput* and *Elev*. The performance of the models decrease when we ablate CVaR with mean. **Best** , <u>Second best</u>.

Ablating CVaR To demonstrate the effectiveness of using CVaR during planning to capture worst-case risk, we ablate it by setting $\alpha = 0$, which is equivalent to using the mean of the predicted distribution. As reported in Table 2, the performance of both our model and *AngleInput* dropped. This is because accounting for worst-case (tail) risk is crucial in safety-critical tasks, and CVaR is a principled way for representing tail risk in robotics [28].

Ablating Maximum Frequency As demonstrated in Section 3.2 and Section 3.3, the choice of maximum frequency n is closely related to the smoothness and generalization of the mapping from angle of approach ϕ to each of the concentration parameters γ^i_ϕ. Therefore, we adjust the number of Fourier bases n in Eq. (3) and analyze the results in this section. We consider the cases $n = 1$ and $n = 5$, and visualize the test EMD^2 loss in Fig. 11. Overall, we observe similar test EMD^2 loss for all three models. In Table 2 we include their performances on the aforementioned experiment, on which they achieve success rate 91% and 94%, respectively. Notice the model with $n = 1$ achieves lower success rate compared to the other two models. We hypothesize this is because the basis functions can be too simple to capture the change of γ^i_ϕ with ϕ, i.e. it might not be expressive enough to parameterize the mapping we are interested in. Although the model with maximum frequency $n = 5$ achieves similar success rate as ours ($n = 3$), an overly high maximum frequency n introduces higher upper bound on the Lipschitz constant of the mapping and can introduce more abrupt changes in γ^i_ϕ with small changes in ϕ, as explained in Section 3.3. Therefore, to balance expressivity and generalization, we choose $n = 3$ in our model and our experiments. However, n is an easily changed hyperparameter that can be adjusted depending on the frequency composition of the mapping function we are trying to approximate.

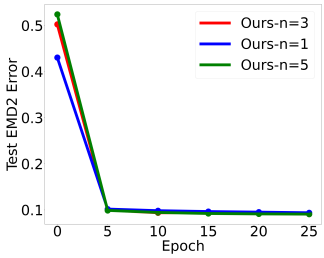


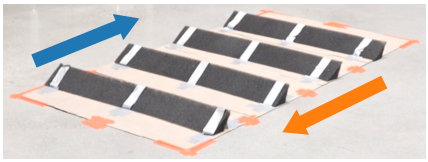
Figure 11. Test EMD^2 loss of our model with varying maximum frequency. They achieve similar test loss.

F Hardware Specification

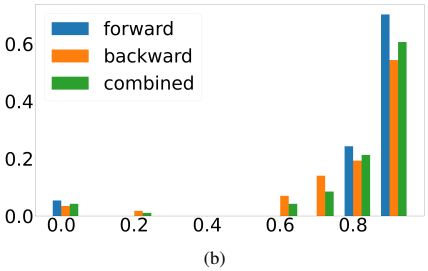
Our robot is built on top of the AgileX Scout Mini chassis, a skid-steering wheeled platform. It has a footprint of approximately $0.6 \times 0.6m$ and can achieve a maximum speed of 3 m/s. The robot is equipped with an Ouster OS1 LiDAR with 128 channels. The LiDAR runs with a horizontal resolution of 1024 at 10 Hz, and we use its internal 6-axis IMU (100 Hz). The point cloud and IMU measurements are processed using DLIO [29] for prebuilding the point cloud map and online localization. The onboard PC is an ASUS ROG NUC 970, with an Intel Core Ultra 9 CPU (16-Core), 32 GB RAM, and a Nvidia GeForce RTX 4070 GPU (8 GB).

G Generalization to Other Problem Settings

In this section, we first demonstrate other problem settings that could benefit from modeling angle-dependency. We use the angle-dependency of linear traction (the robot's ability to execute commanded velocity on the given terrain [4]) as an example, and show how our approach can be applied to this problem setting. To construct an empirical dataset, we command the vehicle to drive at a constant linear velocity of 0.5 m/s over the obstacle, along the forward (Blue) and the backward (Orange) direction shown in Fig. 12a. We record the achieved linear velocity (estimated via DLIO [29]), and compute the linear traction as the ratio between the achieved linear velocity and the commanded linear velocity (0.5 m/s). The data points are discretized into histograms and then normalized into categorical distributions. As shown in Fig. 12b, the backward traversal (Orange) achieved significantly lower linear tractions compared to the forward traversal (Blue). This matches our intuition because the obstacle is smooth from the forward direction and sharp from the other. A system that models the angle-



(a)



(b)

Figure 12. (a): Obstacle and directions of the traversals. (b): The linear traction distribution of the forward (Blue) and backward (Orange) traversal. The backward distribution has significantly more probability mass in the lower bins. An algorithm not modeling the angle-dependency of linear traction (Green) would not be able to identify the obstacle as traversable from the forward orientation.

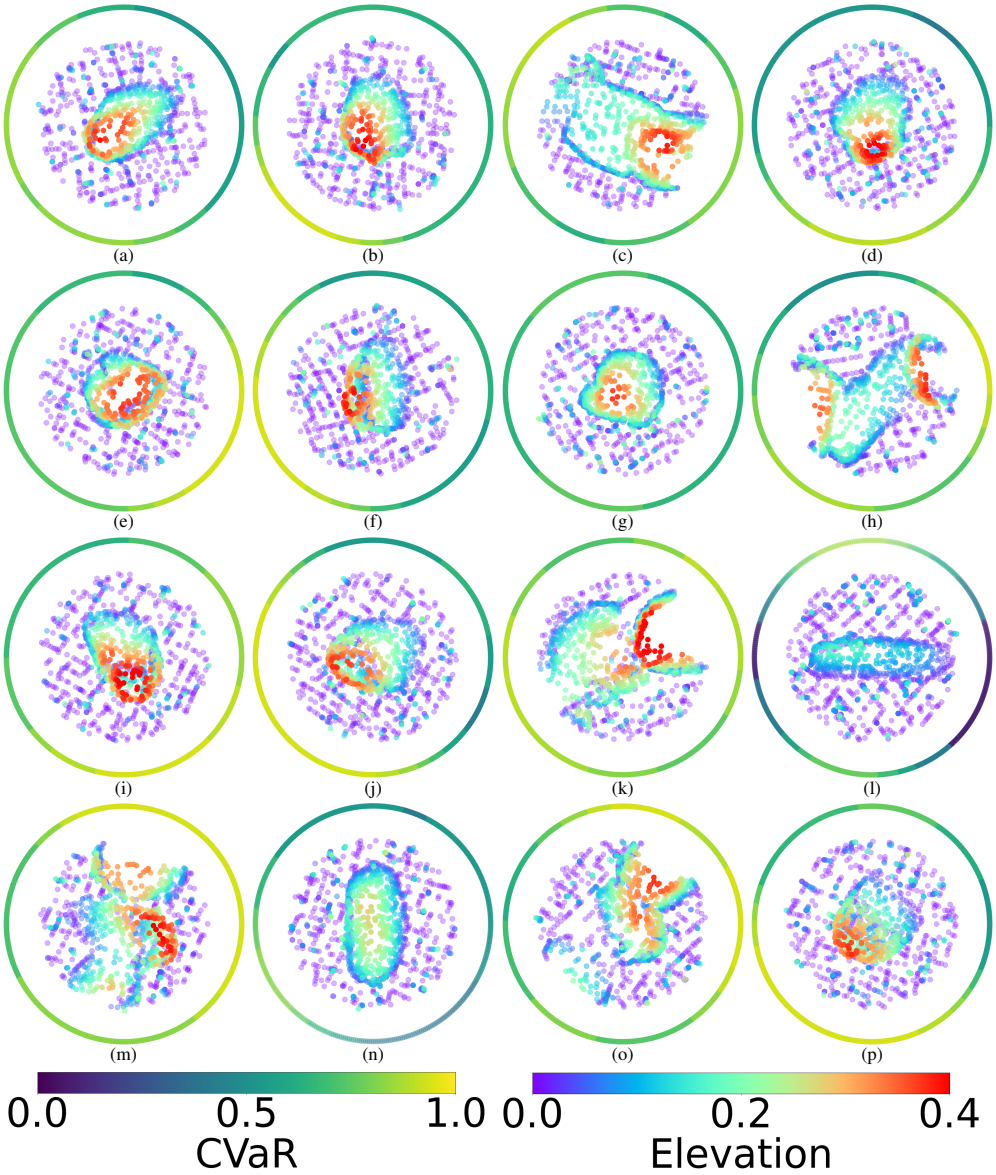


Figure 13. Example predictions of SPARTA. *Rainbow* denotes point elevation, and *Viridis* circles around the point clouds denote the CVaR at the corresponding angle of approach. The CVaR matches our intuition because it indicates the obstacles are more risky from orientations with sharp edges and less risky from orientations with smooth elevation change.

dependency of linear traction would thus be able to capture this difference. Notice SPARTA can be applied to this problem setting with minimal changes (e.g. adjusting input point cloud size) using an empirical dataset of linear traction distributions. However, systems that do not model angle-dependency (Green) [4] could be overly conservative and never command the robot to traverse the obstacle from the forward direction, which is traversable in practice.

In fact, we anticipate any variables that exhibit smooth and continuous behaviors, especially those that depends on variables with Lie group structure, could similarly benefit from our representation framework. **SPARTA** can be easily modified to use their respective basis functions and predict the corresponding coefficients.

H Example CVaR Predictions of SPARTA

In Fig. 13, we visualize some example obstacles in our test set and the CVaR predicted by our model with angle of approach $\phi \in \{0, 1, \ldots, 359\}$. The *Rainbow* color map denotes the elevation of the point (Red denotes high elevation), and *Viridis* denotes the predicted CVaR (Yellow corresponds to

high CVaR). A trend we notice from the examples is that our model predicts lower CVaR for angles where the obstacle has a smoother elevation change. This is aligned with our intuition, further supporting the validity of our approach for identifying risky obstacles and angles of approach.