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 [27], 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 [27]. 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 [27] 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 [27] 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 [6]. In contrast, the runtime of *AngleInput* increases with complexity.

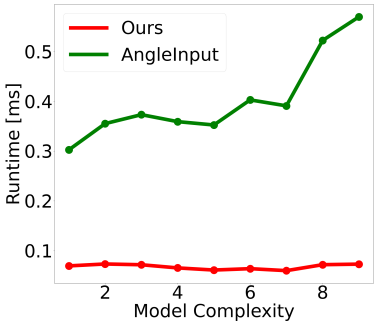


Figure 9. Runtime of *AngleInput* (Green) increases with model complexity, while the runtime of ours (Red) remains roughly constant. This is because during planning, we only need to perform a low dimensional dot product and a sigmoid pass during inference, whereas *AngleInput* requires repeated neural 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

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

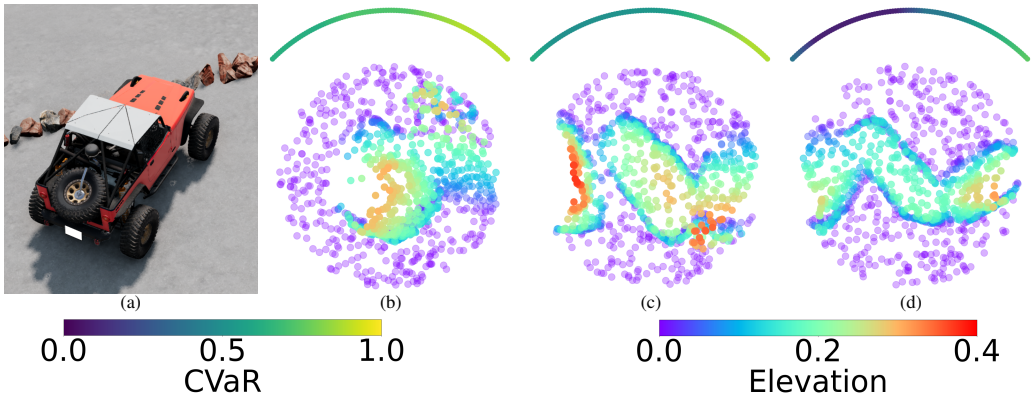


Figure 10. (a): Overview of the test environment and the vehicle. (b)-(d): Example point cloud (colored by elevation) and the CVaR (Viridis arces around the point clouds) predicted by our model at the candidate angles of approach.

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 [25].

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_ϕ^i. 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_ϕ^i 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_ϕ^i 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.

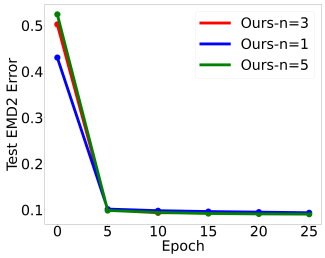


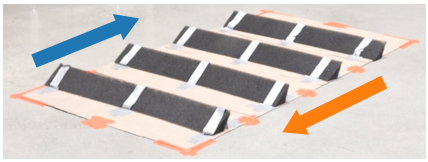
Figure 11. Test EMD^2 loss of our model with varying maximum frequency. They achieve similar test loss.

F Hardware Specification

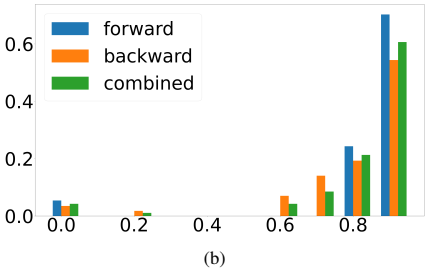
Our robot is built on top of the AgileX Scout Mini chassis, a skid-steering wheeled platform. It has a footprint of approximately $0.6 \times 0.6m$ and can achieve a maximum speed of 3 m/s. The robot is equipped with an Ouster OS1 LiDAR with 128 channels. The LiDAR runs with a horizontal resolution of 1024 at 10 Hz, and we use its internal 6-axis IMU (100 Hz). The point cloud and IMU measurements are processed using DLIO [26] 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 Geometric Structure in Off-road Autonomy

In this section, we 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 [26]), 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-dependency of linear trac-



(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.

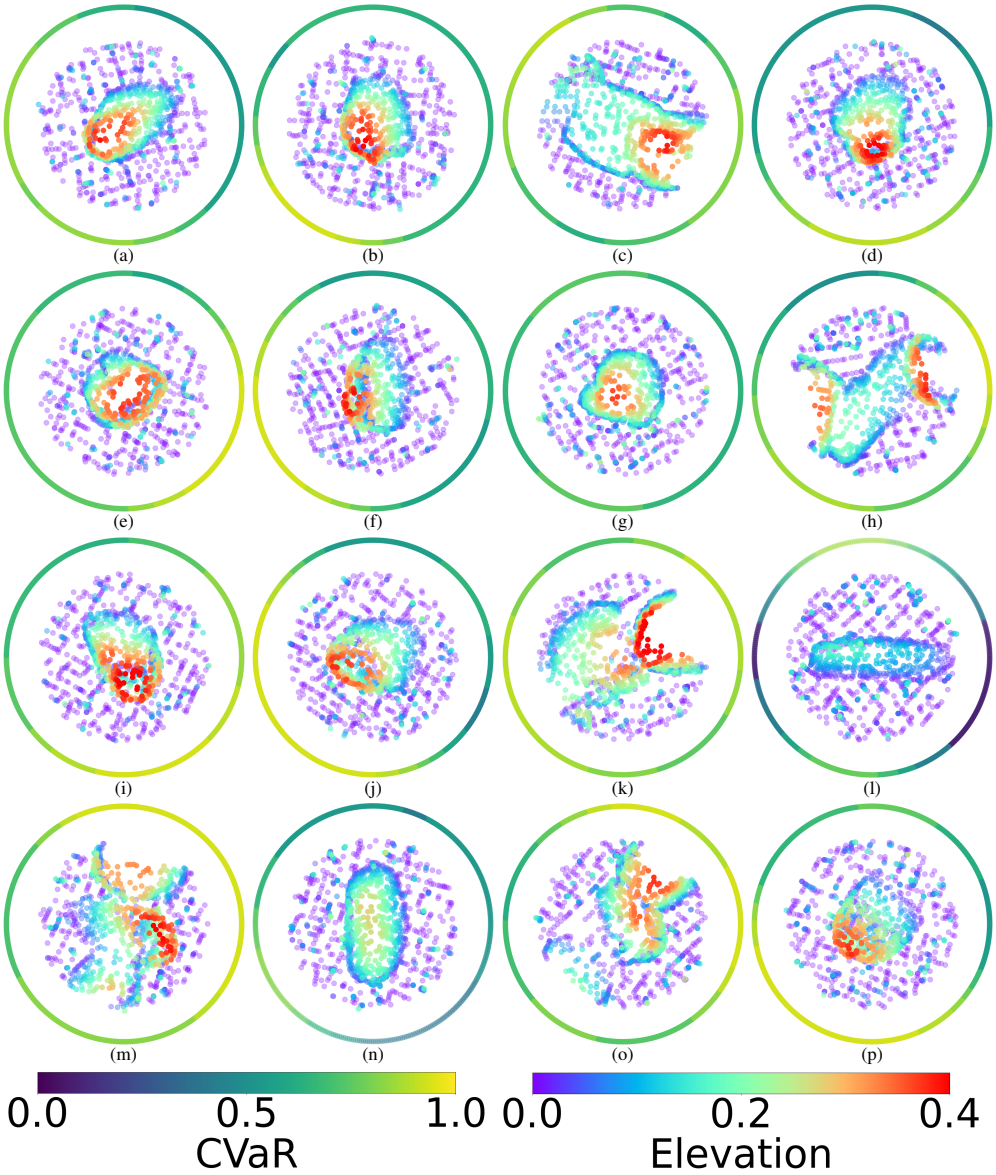


Figure 13. Example predictions of SPARTA. *Rainbow* denotes point elevation, and *Viridis* circles around the point clouds denote the CVaR at the corresponding angle of approach. The CVaR matches our intuition because it indicates the obstacles are more risky from orientations with sharp edges and less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

tion 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.

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.