Semantic Terrain Classification for Off-Road Autonomous Driving Supplementary Material

Anonymous Author(s)
Affiliation
Address
email

Website link: https://sites.google.com/view/terrain-traversability/home

1 Network Architecture

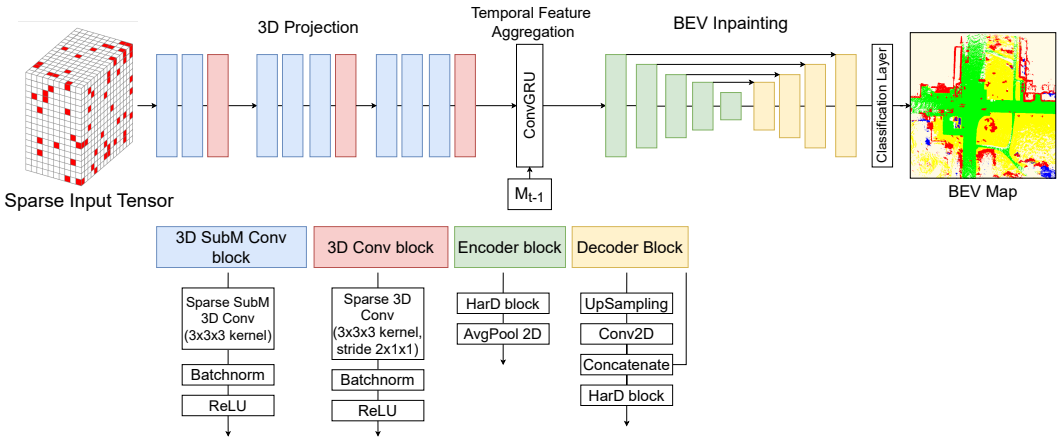


Figure 1: Detailed illustration of our BEVNet-R architecture. BEVNet-S has the same architure with ConvGRU excluded.

We provide a detailed description of our network here, illustrated in Figure 1. The 3D projection is composed of a set of 3D convolution blocks, where the 3D SubM Conv block is a sequence of Submanifold 3D convolution, Bachnorm, and ReLU activation. The 3D Conv block maintains the XY resolution but compresses in the z dimension, and is made up of 3D convolution, Bathnorm, and ReLU activation. The Temporal Feature Aggregation module has a single ConvGRU layer for efficiency. The BEV Inpainting network is a modified variant of FC-HardNet with skip connections based on Harmonic DenseNet [1]. Each encoder block is a Harmonic Dense (HarD) block with Average Pooling, with no pooling for the middle layer. Each decoder block upsamples the features with Transposed convolution with bilinear upsampling, and features from the corresponding encoder block is concatenated. Each decoder block has a HarD block following the skip connection. The final BEV map is predicted with a Fully convolutional layer as the classification layer.

Network training. We train our network using the Adam optimizer [2] with an initial learning rate of $3e-4$ and a decay of 0.7 per epoch. We use a weighted Cross-Entropy loss. We start with training a single-frame model without ConvGRU until the model converges. Then we freeze the sparse convolution layers and insert the ConvGRU layer, and then train the ConvGRU and the inpainting network together. While technically we can train the whole network end-to-end , this two-stage training procedure is faster and is more memory-efficient. When training the ConvGRU, we use a sequence length of 5 with a frame stride randomly chosen from [1, 10, 20]. Training takes about 12 hours on a single RTX 3090. The inference time of our network is 6 fps on a RTX 3090.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

Dataset	free	low-cost	medium-cost	lethal
SemanticKITTI	road parking sidewalk other-ground	terrain	vegetation	car bicycle bus motorcycle on-rails truck other-vehicle person bicyclist motorcyclist building fence trunk pole traffic-sign moving-car moving-bicyclist moving-person moving-motorcyclist moving-on-rails moving-bus moving-truck moving-other-vehicle
RELLIS-3D	dirt asphalt concrete	grass puddle mud rubble	bush	tree pole vehicle object building log person fence barrier

Table 1: Mapping of the original class labels to the 4-level traversabilities for the SemanticKITTI and RELLIS-3D datasets.

Data augmentation. During training, we randomly rotate every pair of LiDAR scans and the ground truth traversability map in $\mathcal{U}[-45°, 45°]$, and randomly drop 20% of the points. Furthermore, we perturb the groundtruth odometry with rotation drawn from $\mathcal{N}(0, 0.01^2)$ and translation drawn from $\mathcal{N}(0, 0.1^2)$. Note that the error in odometry will accumulate over time.

2 Dataset

2.1 Traversability Mapping

Table 1 shows how we map the raw class labels to the 4-level traversabilities.

2.2 Class distribution

Figure 2 shows the class distribution of SemanticKITTI and RELLIS-3D using our BEV labels based on the 4-level traversability ontology.

2.3 Train/Validation Split

SemanticKITTI We choose sequence 08 as our validation sequence, which is typically done by other works including but not limited to which [3] we compare to.

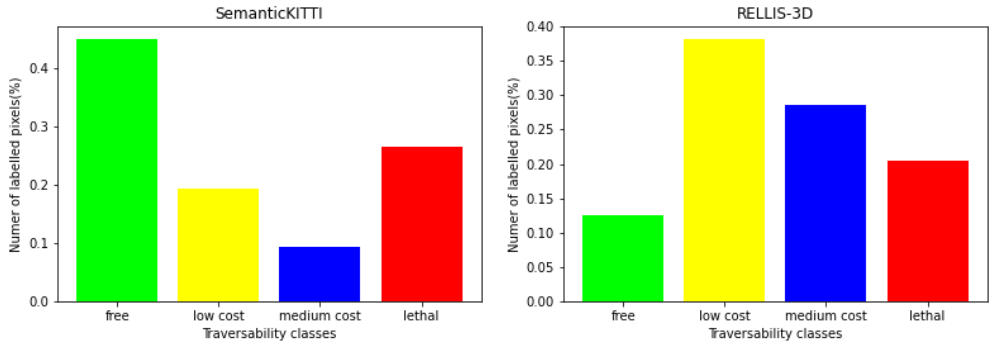


Figure 2: Individual class distribution across our 4-class ontology on SemanticKITTI (left) and RELLIS-3D (right). Distribution is recorded from the total dataset as the proportion of total pixels each class has in the generated BEV labels.

RELLIS-3D We choose sequence 04 as our validation sequence. While the authors of the dataset [4] provide a split, it follows random selection of samples and therefore does not fit our training of recurrent model. Instead, we select a whole sequence as a holdout dataset.

3 More Quantitative Results

We present additional quantitative results here including classwise Intersection over Unions. Moreover, we add an additional model "BEVNet-R (C)" which is trained on clean odometry. The model in the main paper "BEVNet-R" is renamed to "BEVNet-R (N)" (trained on noisy odometry). Specifically:

- Table 2,3,4 show the results on the SemanticKITTI dataset for the "all", "seen", and "unseen" modes, respectively.
- Table 5,6,7 show the results on the RELLIS-3D dataset for the "all", "seen", and "unseen" modes, respectively.

	free	low-cost	medium-cost	lethal	mIoU
BEVNet-S	0.666	0.484	0.113	0.456	0.430
Clean Odometry					
BEVNet-S + TA	0.672	0.516	**0.271**	0.500	0.490
BEVNet-R (N)	0.719	0.551	0.171	0.480	0.480
BEVNet-R (C)	**0.768**	**0.620**	0.215	0.538	**0.535**
Cylinder3D + TA	0.612	0.510	0.264	0.476	0.465
Noisy Odometry					
BEVNet-S + TA	0.578	0.449	**0.232**	0.328	0.397
BEVNet-R (N)	**0.705**	**0.532**	0.171	**0.464**	**0.468**
BEVNet-R (C)	0.669	0.523	0.192	0.451	0.459
Cylinder3D + TA	0.466	0.390	0.193	0.318	0.342

Table 2: SemanticKITTI. Including both the seen and unseen area.

4 Additional Comparison with Various Baselines

Several recent papers have focused on semantic understanding of scenes from sparse LiDAR scans, which we discuss below. These papers solve related, but slightly different problems. They produce smaller maps, and do not perform temporal aggregation.

- **Semantic Scene Completion.** The scene completion task aims to predict a dense semantic voxel grid from a single LiDAR scan. To compare to our work, we build a traversability map by converting predicted dense voxel grids to 2D traversability maps using the same method described in Sec 3.2.

	free	low-cost	medium-cost	lethal	mIoU
BEVNet-S	0.713	0.529	0.148	0.538	0.482
Clean Odometry					
BEVNet-S + TA	0.723	0.571	0.363	0.590	0.562
BEVNet-R (N)	0.776	0.613	0.227	0.572	0.547
BEVNet-R (C)	0.844	0.703	0.298	0.654	0.625
Cylinder3D + TA	**0.864**	**0.745**	**0.378**	0.635	0.655
Noisy Odometry					
BEVNet-S + TA	0.600	0.480	**0.308**	0.362	0.438
BEVNet-R (N)	**0.756**	**0.588**	0.224	**0.549**	**0.529**
BEVNet-R (C)	0.723	0.575	0.247	0.532	0.519
Cylinder3D + TA	0.617	0.532	0.272	0.399	0.455

Table 3: SemanticKITTI. Only the seen area.

	free	low-cost	medium-cost	lethal	mIoU
BEVNet-S	0.560	0.398	0.069	0.247	0.319
Clean Odometry					
BEVNet-S + TA	0.558	0.409	**0.147**	**0.278**	0.347
BEVNet-R (N)	0.597	0.430	0.094	0.255	0.344
BEVNet-R (C)	**0.608**	**0.461**	0.096	0.249	**0.354**
Cylinder3D + TA	-	-	-	-	-
Noisy Odometry					
BEVNet-S + TA	0.522	0.388	**0.129**	0.249	0.322
BEVNet-R (N)	**0.593**	**0.424**	0.098	**0.254**	**0.343**
BEVNet-R (C)	0.553	0.420	0.113	0.244	0.332
Cylinder3D + TA	-	-	-	-	-

Table 4: SemanticKITTI. Only the unseen area.

	free	low-cost	medium-cost	lethal	mIoU
BEVNet-S	0.493	0.550	0.597	0.596	0.559
Clean Odometry					
BEVNet-S + TA	**0.737**	**0.613**	0.544	0.565	0.615
BEVNet-R (N)	0.557	0.602	0.632	**0.682**	0.618
BEVNet-R (C)	0.675	0.593	**0.633**	0.676	**0.644**
Cylinder3D + TA	0.250	0.413	0.398	0.584	0.411
Noisy Odometry					
BEVNet-S + TA	**0.621**	0.522	0.293	0.370	0.452
BEVNet-R (N)	0.553	**0.598**	**0.628**	**0.675**	**0.614**
BEVNet-R (C)	0.266	0.361	0.473	0.450	0.387
Cylinder3D + TA	0.229	0.348	0.289	0.407	0.318

Table 5: RELLIS-3D. Including both the seen and unseen area.

	free	low-cost	medium-cost	lethal	mIoU
BEVNet-S	0.230	0.578	0.617	0.649	0.518
Clean Odometry					
BEVNet-S + TA	**0.611**	0.638	0.560	0.609	0.605
BEVNet-R (N)	0.323	0.622	0.650	0.711	0.577
BEVNet-R (C)	0.489	0.621	0.655	0.716	0.621
Cylinder3D + TA	0.478	0.660	0.509	**0.727**	0.568
Noisy Odometry					
BEVNet-S + TA	0.398	0.514	0.225	0.350	0.372
BEVNet-R (N)	0.322	**0.617**	**0.646**	**0.705**	**0.572**
BEVNet-R (C)	0.07	0.327	0.462	0.457	0.329
Cylinder3D + TA	**0.460**	0.438	0.357	0.485	0.435

Table 6: RELLIS-3D. Only the seen area

	free	low-cost	medium-cost	lethal	mIoU
BEVNet-S	0.713	0.488	0.546	0.433	0.545
Clean Odometry					
BEVNet-S + TA	**0.843**	0.555	0.502	0.446	0.586
BEVNet-R (N)	0.754	**0.559**	**0.583**	**0.584**	0.620
BEVNet-R (C)	0.830	0.530	0.574	0.547	**0.621**
Cylinder3D + TA	-	-	-	-	-
Noisy Odometry					
BEVNet-S + TA	**0.811**	0.546	0.460	0.421	0.560
BEVNet-R (N)	0.748	**0.557**	**0.579**	**0.579**	**0.616**
BEVNet-R (C)	0.435	0.432	0.509	0.420	0.449
Cylinder3D + TA	-	-	-	-	-

Table 7: RELLIS-3D. Only the unseen area.

- **Inpainting Network.** Han et al. [5] recently proposed an approach that uses GANs to inpaint a sparsely segmented BEV image. We trained our model using the same SemanticKITTI dataset with the groundtruth 19-class BEV images as supervision. Note that this task is different from traversability estimation because it does a simple topdown projection.

Implementation Details. For a fair comparison, we re-train our models using the settings where the baselines were trained on: the map size is $51.2m \times 51.2m$, and the vehicle is located at the center left. To have a fair comparison, for each frame we aggregate Cylinder 3D predictions over the past 70 frames using the calibrated poses provided by SemanticKITTI and do a top-down projection by picking the highest z point at each location. JS3C-Net is a 3D scene completion algorithm. We use the model and the code provided by the authors. We take the 3D voxel prediction of JS3C-Net and project it to 2D using top-down projection. The result of Han et al. [7] is reported from the original paper since the output is already in 2D.

Results: The results are shown in Table 8. When testing on full categories and using top-down projection, JS3C-Net is slightly better than our method, specifically for classes within the lethal obstacles group. Note that in our approach, the projection is learned from the data and not hand-engineered as in top-down projection. We hypothesize for simple top-down projection, it is more data-efficient to code the projection method instead of learning it from the data. However, in complex projection, our learning-based approach in the 4 category ontology significantly outperforms JS3C-Net.

Speedwise, JS3C-Net is significantly slower than ours since it predicts a dense voxel grid. On a 51.2m x 51.2m map, JS3C-Net runs less than 2 fps whereas BEVNet-S runs at 12 fps on a 1080 Ti. It will be hard to scale JS3C-Net to larger maps with recurrency (note that in our main results we use 102.4m x 102.4m maps). In comparison, BEVNet maintains a latent 2D feature map for cost reasoning, which is more memory-efficient.

Cylinder 3D's predictions are limited to the seen area of the map. It is surprising that our method performs as well as the Cylinder 3D on the seen area while being able to inpaint the entire map. Finally, we emphasize that our network is designed to perform multiple tasks simultaneously (semantic segmentation, inpainting, complex projection, time aggregation). It is not surprising that its performance does not exceed Cylinder-3D or JS3C-Net which perform a subset of these tasks using a network with a similar capacity.

5 Ablation on Odometry Noise

We vary the odometry noise level by introducing a scalar λ such that the rotation noise is drawn from $\mathcal{N}(0, (0.01\lambda)^2)$ and translation noise drawn from $\mathcal{N}(0, (0.1\lambda)^2)$. The results are presented in Table 9. BEVNet-R degrades gracefully as noise level increases. Please check out the project website for qualitative videos on this.

Method (#classes)	Segmentation	Inpainting	Projection	Temporal Aggregation	seen	unseen
Cylinder3D-TA (19)	Yes	No	Top Down	Yes	**0.316**	0.181
Han et al. (19)	No	Yes	Top Down	No	-	0.131
JS3C-Net (19)	Yes	Yes	Top Down	No	-	**0.258**
BEVNet-S (19)	Yes	Yes	Top Down	No	0.313	0.253
JS3C-Net (4)	Yes	Yes	Complex	No	-	0.549
BEVNet-R (4)	Yes	Yes	Complex	Yes	-	**0.608**

Table 8: Comparing with various semantic segmentation and completion baselines on the SemanticKITTI dataset. Note that the map here is smaller than our main results to accommodate the baselines. We evaluate on two kinds of projection: top-down projection (only looking at the highest point at each location), and complex projection (taking both the traversability of each class and their heights into account).

	λ				
	0%	50%	100%	200%	500%
SemanticKITTI	0.480	0.474	0.468	0.458	0.431
RELLIS-3D	0.618	0.616	0.614	0.611	0.600

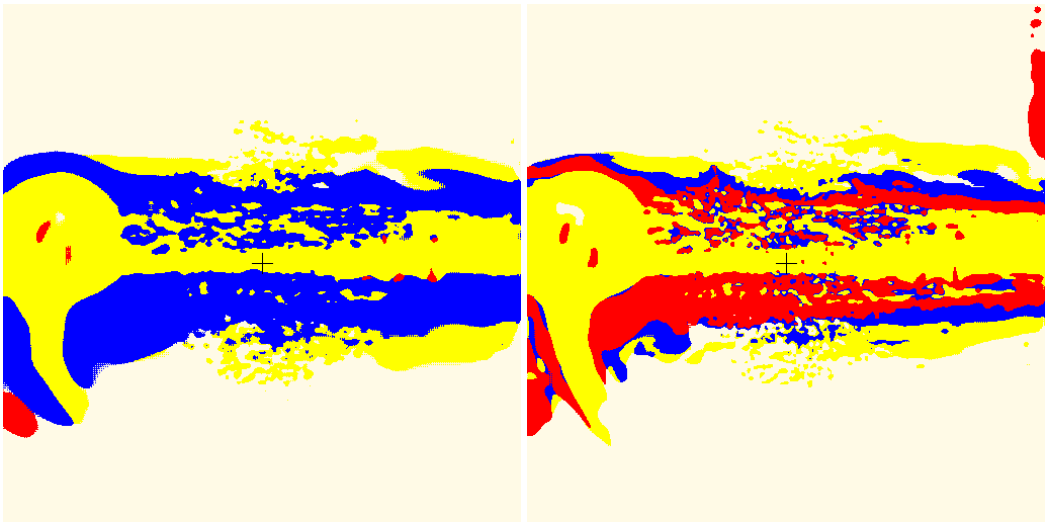
Table 9: Effect of odometry noise level on the mIoU. Note that BEVNet-R is trained at 100% noise level.

6 Robot-dependent Traversability

We show that BEVNet can predict traversability by reasoning both semantic and geometric properties of terrains. We generate two groundtruth BEV costmap datasets from the RELLIS dataset for this experiment.

- The first dataset maps *bush* to the medium-cost class regardless of the height of the bush. This applies to a large offroad vehicle like the ClearPath Warthog.
- The second dataset maps *bush* to the lethal class if the bush is 0.5 m above the ground. This applies to a small offroad vehicle like the ClearPath Husky.

We train BEVNet on each dataset separately. Figure 3 shows the prediction results on the RELLIS-3D validation set. BEVNet is able to predict robot-dependent traversability maps. In Figure 3b, some bushes are labeled as lethal since they are too high and are thus dangerous for a small offroad vehicle.



(a) Predicted traversability map for a large robot. (b) Predicted traversability map for a small robot.

Figure 3: BEVNet can reason both the semantic and geometric properties of the terrain.

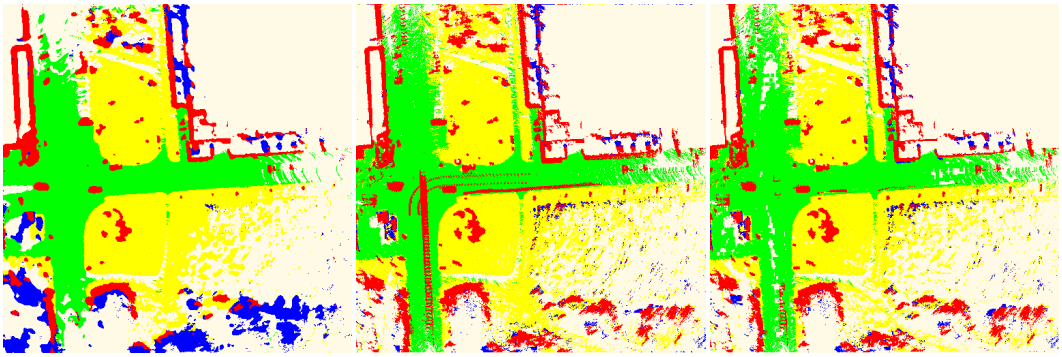
7 Variations of Cylinder3D with temporal aggregation

For a more comprehensive comparison, we additionally compare our method against a more traditional 3D SLAM-fusion based fusion implementation (denoted as Cylinder3D-TA-3D). The 3D baseline performs the following:

1. We first perform semantic segmentation on the LiDAR scans with Cylinder3D to generate segmented point clouds.
2. We aggregate the segmented point clouds in 3D using a voxel grid similar to how Octomap [6] works using the poses that are calibrated by a standard SLAM algorithm.
3. When aggregating the point clouds, we optionally apply ray tracing to remove the trace of moving objects as much as possible.
4. For the points falling into a voxel, we normalize the counts of their predicted labels into a categorical distribution. This is how Bayesian statistics estimates the parameter of a categorical distribution via a uniform Dirichlet prior. This produces a labeled voxel grid.
5. We project this labeled voxel grid down to 2D using the same rule as how we generate the groundtruth BEV costmaps.

7.1 Nuances on comparing the methods on the seen areas.

Depending on how aggregation is performed, the seen areas vary slightly between Cylinder3D-TA, Cylinder3D-TA-3D (w/o ray tracing) and Cylinder3D-TA-3D (with ray tracing) (Figure 4). In particular, when turning on ray tracing for Cylinder3D-TA-3D, some of the moving objects will get cleared, but some ground voxels may also get cleared due to discretization errors (this happens when a LiDAR ray hits the ground with a shallow incident angle). To make a fair comparison, in Table 10 and 11 we report mIoUs of the methods on the three seen areas generated by Cylinder3D-TA, Cylinder3D-TA-3D (w/o ray tracing) and Cylinder3D-TA-3D (with ray tracing).



(a) (b) (c)

Figure 4: Comparing the seen areas of (a) Cylinder3D-TA, (b) Cylinder3D-TA-3D w/o raytracing and (c) Cylinder3D-TA-3D with raytracing. The seen areas are slightly different depending on how aggregation is performed. For a fair comparison, we evaluate the performance of the baselines on the three different kinds of seen areas.

		Mask used for the seen area		
Method	Full	Cy3d+TA	TA-3D w/o ray tracing	TA-3D w/ ray tracing
Cy3D + TA	0.465	**0.655**	0.613	0.618
Cy3D + TA-3D w/o ray tracing	0.482	0.636	**0.660**	0.660
Cy3D + TA-3D w/ ray tracing	0.471	0.629	0.646	**0.677**
BEVNet-R	**0.535**	0.625	0.613	0.615

Table 10: Comparison of different baseline variants across different masks on the SemanticKITTI dataset. Full refers to the whole map (seen + unseen).

Method	Full	Mask used for the seen area		
		Cy3d+TA	TA-3D w/o ray tracing	TA-3D w/ ray tracing
Cy3D + TA	0.411	0.568	0.573	0.567
Cy3D + TA-3D w/o ray tracing	0.408	0.576	**0.649**	0.643
Cy3D + TA-3D w/ ray tracing	0.384	0.539	0.609	**0.645**
BEVNet-R	**0.644**	**0.621**	0.618	0.615

Table 11: Comparison of different baseline variants across different masks on the RELLIS-3D dataset. Full refers to the whole map (seen + unseen).

7.2 Analysis

From the comparisons we observe the following:

- Compared to Cylinder3D+TA baseline that aggregates in the 2D BEV space, 3D aggregation with raytracing exhibits considerable improvement in the seen area: +8 points in RELLIS (+5.9 points in SemanticKITTI) (see the last column of Table 10 and 11), respectively. However, while 2D aggregation time in TA is negligible, 3D aggregation is the computational bottleneck in the pipeline as it takes approx. 1 second to aggregate a single scan into octomap and perform raytracing.
- When evaluating on the whole map, BEVNet outperforms Cylinder3D-TA-3D by more than 20 points in RELLIS (5 points in SemanticKITTI) because Cylinder3D-TA-3D does not predict the future(see the first column of Table 10 and 11).
- When limiting the comparison to the seen area, Cylinder3D-TA-3D outperforms BEVNet-R up to 3 points in RELLIS (6 points in SemanticKITTI) (see the last column of Table 10 and 11).

In conclusion, there is a trade-off between prediction area and accuracy. While combining state-of-the-art semantic segmentation with SLAM-based fusion produces better results on the seen area, it falls short of predicting a more complete map, is prone to odometry noise, and is computationally expensive. Our network is optimized to perform multiple tasks simultaneously (semantic segmentation, inpainting, complex projection, time aggregation) efficiently in a single forward pass.

8 More Qualitative Results

We provide full visualizations of our experiment and ablation study as a set of videos on https://sites.google.com/view/terrain-traversability/home.

References

[1] P. Chao, C.-Y. Kao, Y. Ruan, C.-H. Huang, and Y.-L. Lin. Hardnet: A low memory traffic network. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3551–3560, 2019. doi:10.1109/ICCV.2019.00365.

[2] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[3] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. *arXiv preprint arXiv:2011.10033*, 2020.

[4] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli. Rellis-3d dataset: Data, benchmarks and analysis, 2020.

[5] Y. Han, J. Banfi, and M. Campbell. Planning paths through unknown space by imagining what lies therein. In *CoRL*, 2020.

[6] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.