TerrainNet: Visual Modeling of Complex Terrain for High-speed, Off-road Navigation Appendix

Xiangyun Meng[1], Nathan Hatch[1], Alexander Lambert[1], Anqi Li[1], Nolan Wagener[2], Matthew Schmittle[1], JoonHo Lee[1], Wentao Yuan[1], Zoey Chen[1], Samuel Deng[1], Greg Okopal[1], Dieter Fox[1], Byron Boots[1], Amirreza Shaban[1]
[1]University of Washington [2]Georgia Institute of Technology
https://sites.google.com/view/visual-terrain-modeling

I. Computing Terrain Representation

This section provides more details about how we build the terrain representation.

Point cloud aggregation. Given the raw LiDAR packets and the vehicle's ego motion, we convert the LiDAR packets into LiDAR points with the rolling shutter effects corrected. Then, we run Google Cartographer [1] on the LiDAR points (with IMU information) to compute the gravity-aligned pose for each LiDAR scan. We aggregate the LiDAR scans using their poses to get an aggregated point cloud.

Outlier removal. LiDAR may produce false returns due to the presence of dust, snowflakes, and water. These false returns manifest as noise in the aggregated point cloud. We identify these outliers in two ways:

1) Voxel filter. We voxelize the point cloud with a voxel size of 0.3m. We count the number of points in each voxel. If the number of points is smaller than a threshold (set to 5), we remove all the points in this voxel.
2) Semantic filter. When we label the LiDAR points, we ask the labelers to label the outliers as an additional *outlier* class. Our LiDAR segmentation network is trained with the additional outlier class. After we predict the pseudo labels for each point, we remove points classified as outliers.

II. Network Architecture

Backbone. Our image backbone follows that of Lift-Splat-Shoot[2]. There are two main differences:

1) For RGB-D inputs, we duplicate the first five convolutional blocks to process the RGB and depth images separately (Figure 1). This has a noticeable improvement over passing a 4-channel RGB-D image directly to the backbone.
2) We do $8\times$ downsampling of the feature map and depth map (the original paper performs $16\times$ downsampling). This provides a good trade-off between speed and accuracy. Note that all the baselines are trained with the same image backbone with the same $8\times$ downsampling ratio.

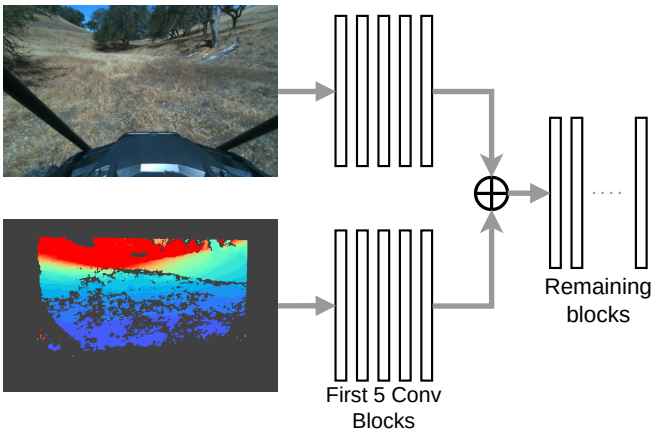

Fig. 1: The image backbone uses separate branches to process the RGB and depth inputs, respectively. Then, the outputs of the two branches are summed and passed to the remaining blocks in the backbone.

Our input image size is 512×320. Hence, the image backbone produces a 472-channel feature map with a spatial dimension of 64×40.

Depth completion network. The output of the image backbone is passed to a single convolution layer with 472 input channels and D output channels with kernel size 3. D is the number of desirable depth bins. For $50\ \mathrm{m} \times 50\ \mathrm{m}$ maps, $D = 128$. For $100\ \mathrm{m} \times 100\ \mathrm{m}$ maps, $D = 256$.

Terrain embedding network. For each back-projected point (x, y, z), we first apply an MLP with layer sizes $(1, 64, 32)$ to z to get the 32-dimensional elevation embedding f_{elev}. Then, we concatenate f_{elev} with the 472-channel image feature f_{sem} and apply another MLP with layer sizes $(504, 96)$ to get the 96-dimensional terrain embedding f. ReLU is applied after each MLP layer.

Temporal aggregation layer. The temporal aggregation layer is a single ConvGRU layer with 96 input channels, 96 output channels, and a kernel size of 1. To train TerrainNet-TA, we first train the non-recurrent version, and then insert the recurrent layer. We freeze the weights of the model except the recurrent layer.

Inpainting network. The inpainting net follows the architecture of Lift-Splat-Shoot's BEV encoder network. The main difference is that we create a separate decoding head consisting of an upsampling layer and two convolution layers to predict a specific terrain layer.

III. Data Annotation

We create our data annotation tool (Figure 2) to obtain ground truth semantically labeled point clouds. We aggregate 30 to 50 LiDAR scans for each selected frame to get a dense point cloud for labeling. We provide camera images for every LiDAR scan for reference so that a labeler can cross-reference to verify that the labels are correct.

IV. Additional dataset examples

Figure 3 shows additional dataset examples to highlight the diversity of the datasets. We collected our datasets from 3 distinctive geographic locations across the year.

V. More qualitative results

Figure 4 shows other example predictions of TerrainNet and other baselines on the validation sets. Figure 5 visualizes different predicted 3D terrains of TerrainNet-TA on the validation sets. For more details, including videos, please visit the website.

References

[1] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2D LIDAR SLAM. In *International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016.

[2] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3D. In *European Conference on Computer Vision*, 2020.

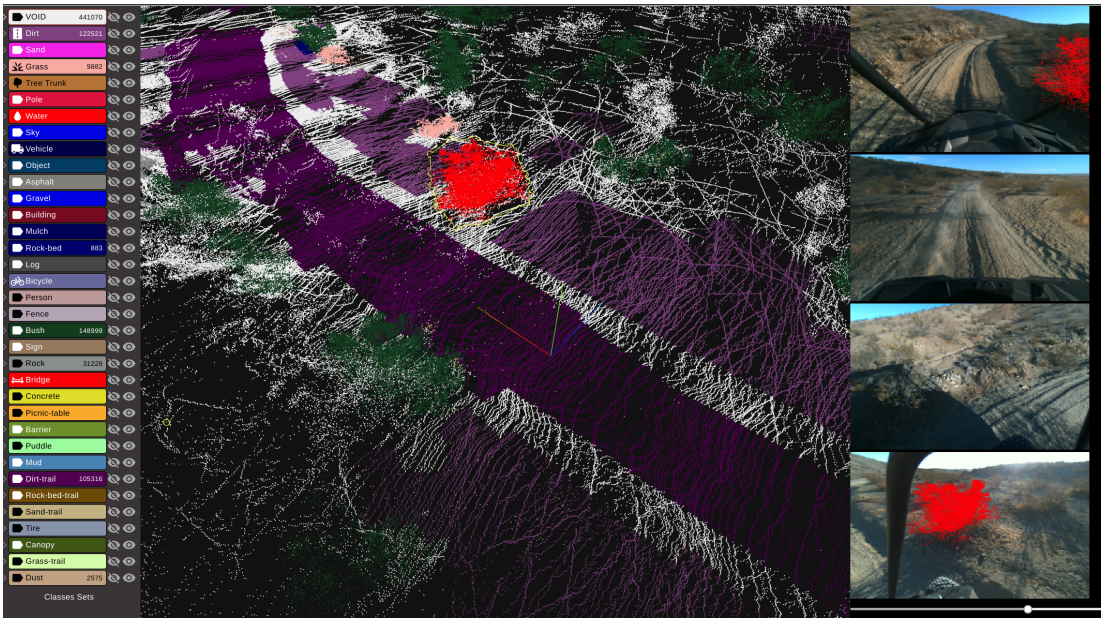

Fig. 2: **Annotation tool.** Labelers can select points in the point cloud and see their corresponding image projections. Here we show a bush is highlighted both in the point cloud and in the bottom image. This helps the labelers to verify the semantic class of those points. Moreover, labelers only annotate points that they are confident about. We leave points that are difficult to identify unlabeled.

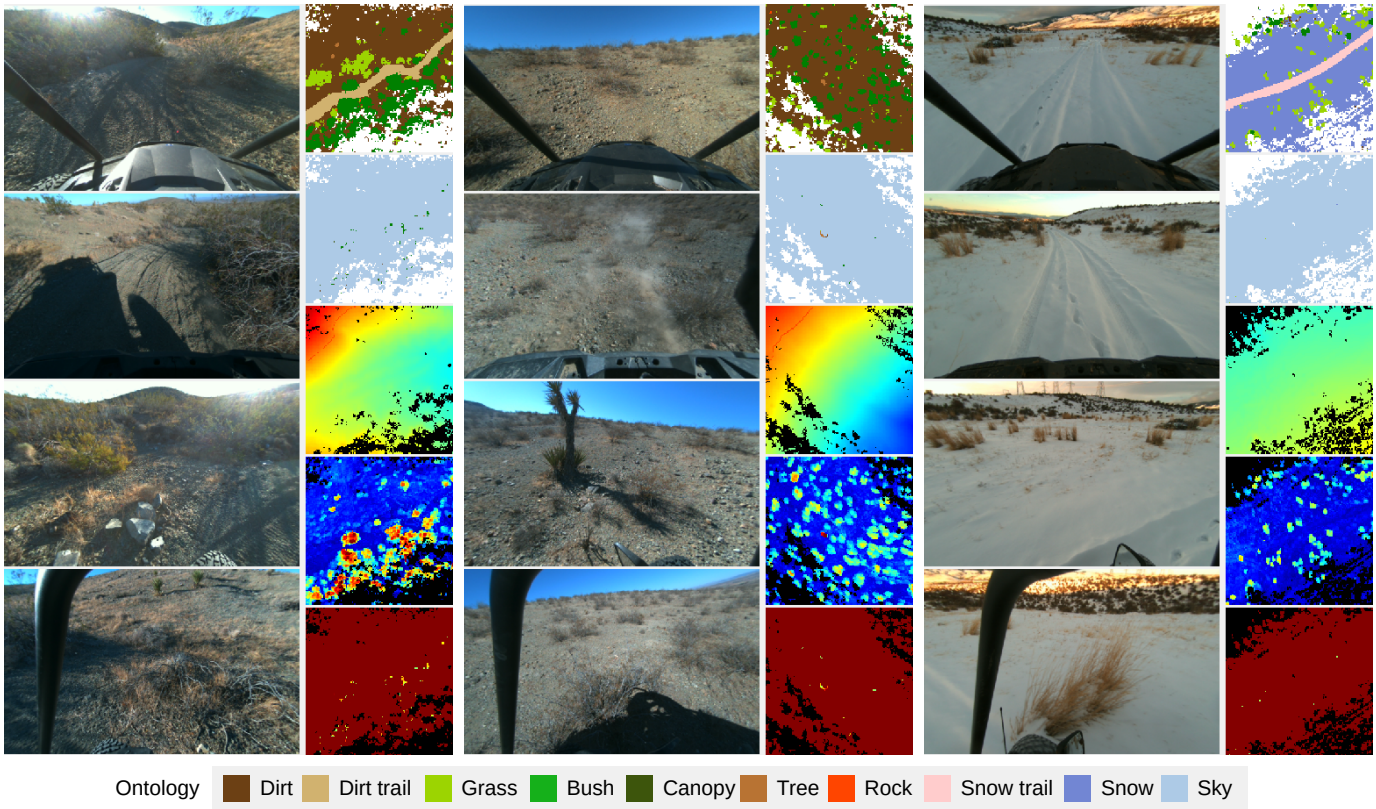

Fig. 3: Additional dataset examples. These examples were collected at different locations and in different seasons. For the real-world experiments, we finetuned the model with annotated snow data (rightmost column).

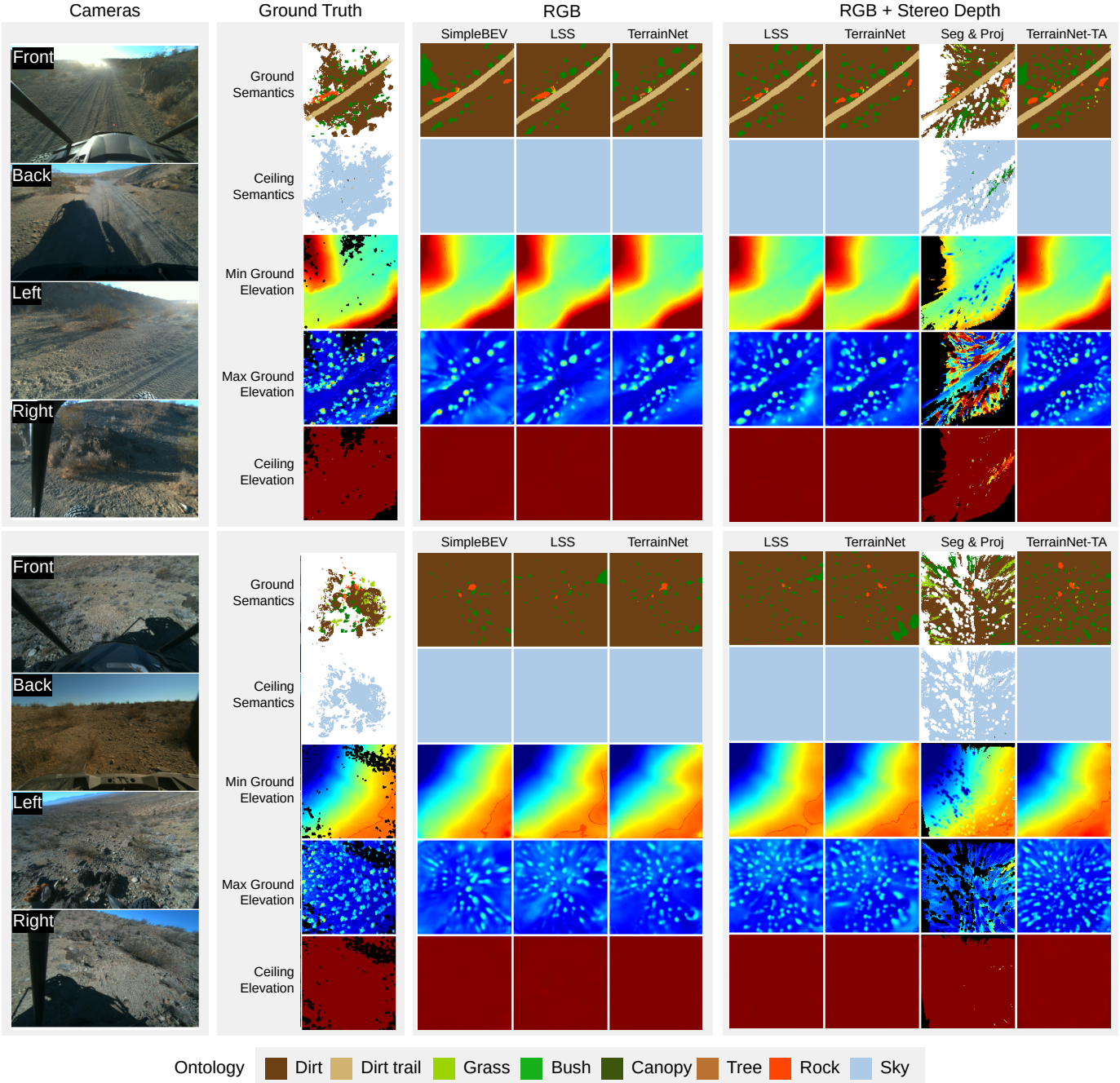

Fig. 4: Comparing the predictions of different models. **Top:** on-trail run with rocks and bushes on the two sides of the trail. **Bottom:** off-trail run with scattered rocks and bushes. These are open terrains, so the ceiling semantics consists almost entirely of *sky*.

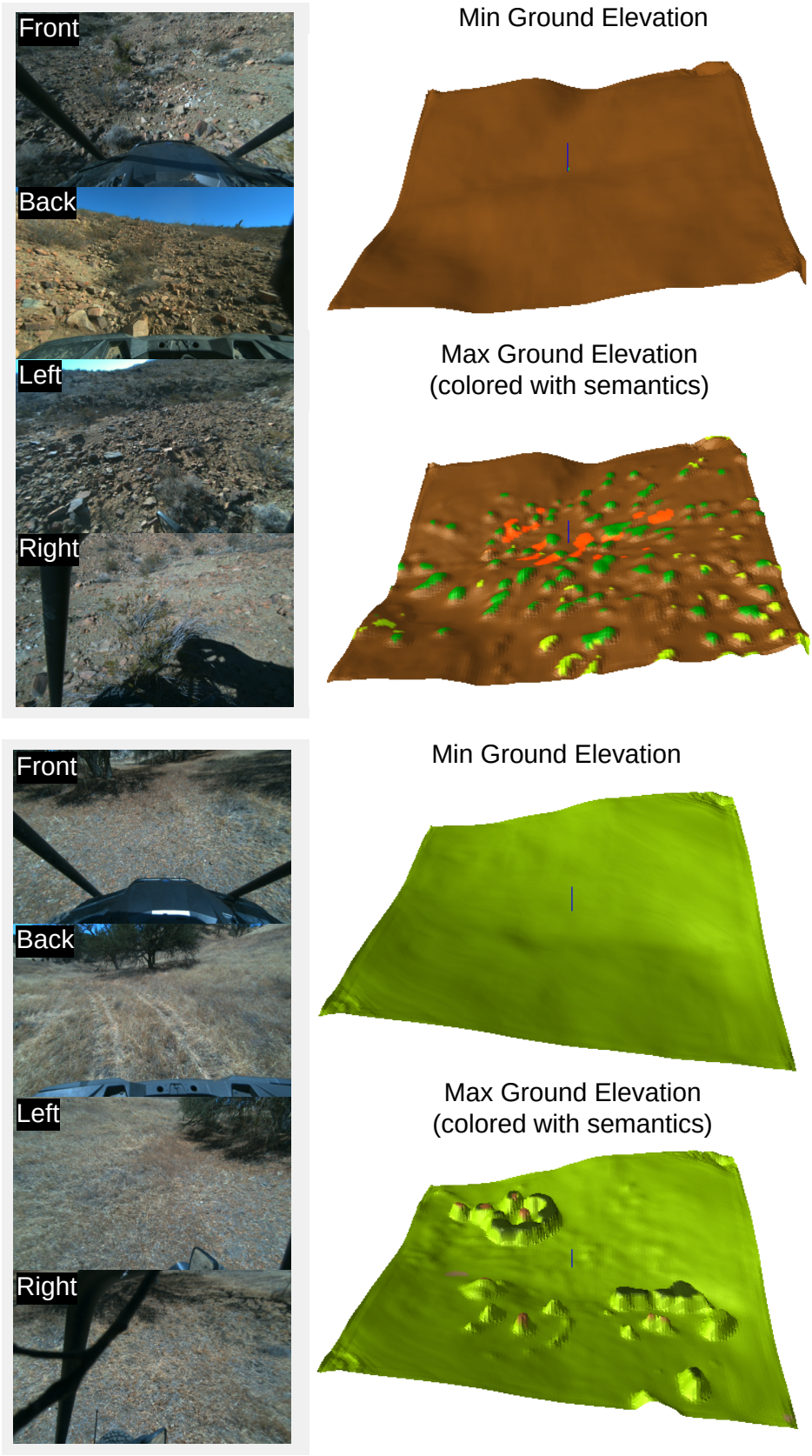

Fig. 5: **Visualizations of the predicted 3D terrain model from TerrainNet-TA**. We only show the min and max ground elevation here for clarity. Left: uneven terrain with scattered grass, bushes, and rocks. Right: a grass-covered valley with tall trees and low-hanging canopies.