Jaskrit Singh
School of Robotics Engineering
Worcester Polytechnic Institute
Worcester, Massachusetts
jsingh3@wpi.edu

I. Motivation

Navigating in unfamiliar, offroad terrain is a challenging problem. It is difficult to tell where a vehicle can traverse, and given two traversable paths it is difficult to tell (algorithmically) which path is better to take. The central focus of this paper will be developing a traversability model of the environment so that we can better answers to these questions.

Once we have a model for the traversability of the environment, we can use it to generate a traversabilty map of the environment. This map can then be used with a planner to plan safer paths, choosing terrain that is easier to traverse. Thus, the primary measure of a good traversability map is that it allows a planner to produce low cost paths.

There are a number of challenges that make offroad traversability mapping a tough problem. Unlike driving on roads, offroad terrain is unstructured; we can't just follow a road that we know to be traversable. Furthermore, there are many different kinds of terrain and it is infeasible that we can collect driving data for our vehicle of interest in all the terrain types we expect to encounter. So a pressing question is how to make a model that generalizes to unseen terrain types. Finally, traversability is vehicle specific. Given a terrain feature, like a ditch, some vehicle may be able to cross it while others would get stuck. This means that we need to consider how traversability models can transfer to different vehicles.

II. Related Work

We must consider several design choices for our traversability mapping algorithm. We can take inspiration from related work and choose the method that fits best.

A. Traversability Model

First, we must choose a traversability model. Some papers use binary classes [2] or split terrain into 3 classes: safe, risky, and obstacle [5]. Other papers use a scalar model, where each point in a grid is assigned a score from 0 to 1 [4] [7] [8] [9] [11]. Still others have a directional model where traversability is different in different directions [6] [10]. Finally, for the most granular modeling of traversabilty, some papers split traversabilty into different metrics such as bumpiness and locomotion [10].

Using a binary or 3 class model for traversabilty can work for simple environments with defined paths and obstacles. This model matches the paradigm used by many motion planners, making it easy to integrate with them. However, offroad traversability generally cannot be broken down into 3 classes. More classes are needed to describe the difference between an asphalt road, an offroad trail, grassy hills, gravel paths, mud, and other offroad terrain.

Scalar traversability offers a little more expressive power while keeping the model simple enough that it is still simple to integrate with a planner. The main limitation of this approach is that we do not model directional effects. When creating a local map, such as in [4] [7] [9] [11] orientation is implicitly added to the model based on the orientation of the vehicle. However, when creating a global map, as we will do in this paper, not modeling directional effects removes important information for offroad environments such as different traversabilities for going uphill vs downhill.

Directional traversability [6] [10] is able to capture the effects of hills and can also aid a planner in charting a path through difficult terrain that must be hit at a certain angle. However, when computing directional traversability one must consider the angular resolution to use. Too high of an angular resolution could result in an increase in storage and computation costs. Furthermore, planners are generally not designed to be used with a directional costmap, so integrating with planners requires additional work.

Finally, for the most fine-grained traversability model, [10] splits traversability into separate metrics: locomotion, energy consumed, and bumpiness. The paper argues that these separate metrics can be weighted at runtime based on the situation. Decreasing locomotion is associated with a higher difference between predicted and actual motion. Meanwhile, energy consumed can be calculated using the power exerted by the wheels over a period of time and the distance traveled. Lastly, bumpiness is measured in different ways between the different papers, but for this paper we will use the idea from [7] that bumpiness is tied to the vertical acceleration signal in a 1-30 Hz range. When modeling traversability it is important to consider which traversability metric is being used and what the metric measures.

Since we are creating a global map of offroad terrain in this paper we will need to use directional traversability constrained to between 4 and 8 directions at each point. This should strike a balance between complexity and an accurate model. If needed for a certain planner, we can always average the different directions to return back to a scalar traversability model. For simplicity, we will not store multiple traversability metrics, only focusing on bumpiness as that is the easiest to measure.

B. Traversability Measurement

Now that we now how we are going to model traversability, we must consider how we are going to measure it.

The simplest method used by [1] [2] [3] is to use geometric features. These papers use LIDAR to create a pointcloud of their immediate environment, then calculate geometric features based on the pointcloud such as step height, slope and roughness. These features can then be used to measure traversability, either directly as in [1] [3] or through an ML model as in [2]. Geometric methods are simple, generalizable to new environments, and catpure directional effects. However, they cannot model different surface types such as mud and tall grass. They also need to be converted from features into an actual traversability score.

Papers [2] and [3] overcome some of the downsides of the geometric features by combining them with semantic information. They use a network that can map pictures to traversability values, such as roads are traversable and puddles are less traversable. They can then fuse the insights from both models to get a robust traversability value. The main issue with segmentation is that we need a mapping from a segment to traversability which generally requires hand labeling. The other limitations are that segmentation does not offer fine-grained resolution (the whole segment gets a single traversability score) and the traversability is not directional.

Another paper [4] takes a different approach to segmentation. Given offline data, it first records where the vehicle drove over and assigns these pixels on each image as being traversable. Next it uses the off-the-shelf Segment Anything Model (SAM) to segment the image from the front camera. This segmentation is then used to inflate the traversable region of each image to include the whole path that the vehicle is driving on. This way they were able to train an image model to detect traversability. While the final model was able to predict a visually coherent scalar traversability map, it would seem that the lack of physical feedback and the lack of partially traversable training data should limit the abilities of this model in certain circumstances such as out of distribution examples.

A different approach used by [6] [7] [8] [9] [10] is to use a self supervised approach using IMU feedback. Generally these approaches use camera and/or IMU data to gather data in front of the vehicle. Later, when the vehicle goes over a patch they saw earlier, they record a traversability metric based on IMU feedback. Different papers use different metrics, but the two most common were the vertical acceleration signal and predicted vs actual motion. Predicted vs actual motion was used by both papers that trained in simulation, likely due to its ability to model crashing into obstacles. Since crashing into obstacles is ill-advised in real world testing, the other papers focus on vertical acceleration signal and similar metrics.

This self supervised method is powerful in that it does not require labeled data, the framework works with different vehicles and it produces oriented predictions. However, one limitation is that it requires driving over a patch to get a traversability measure for it, which is not always possible. The second limitation is that their is no way to directly transfer the model to a new vehicle, it must be retrained, or at least fine-tuned.

This paper will work off of the self supervised approach used in [7]. Patch features will include RGB data and geometric features, such as height, roughness, and slope. Finally, since driving into obstacles does not produce a vertical acceleration, we will use geometric features to directly model obstacles.

C. Data Collection

Papers [6] and [10] collected driving data in simulation while many other papers used real world data, either that they collected themselves, or data available through an offroad driving dataset such as [12].

Using real world data avoids the sim2real gap, with realistic physics, sensors, and cameras. However, its main limitation is that it contains little data from less traversable regions since it can be uncomfortable or even dangerous to drive over these regions. This limits the resolution of the less traversable end of our predictions. Additionally, offroad driving datasets tend be be small, with [12] only having 7 hours of driving data from one region. This limits the model size and generalizability.

Another limitation of real-world datasets is that they are limited to the vehicle they were collected on. This is a serious limitation since, in general, traversability maps cannot directly be transferred between vehicles with different dynamics.

Collecting data in simulation has the advantage that the vehicle can be sent into dangerous regions, getting filling out data on the less traversable end of the spectrum. It also has the advantage that a lot of data can quickly be collected from many different environments.

The main limitation of collecting data in simulation is the Sim2Real gap. While simulators like CARLA have realistic visuals and simulators like Chrono have realistic vehicle physics, there is no simulator that achieves both while running efficiently.

For this paper we collect data in simulation. Using simulation, we are able to easily test differences in traversabilty between vehicles and can gain data in dangerous situations.

III. Methodology

Our methodology closely follows [7]. We drive our car in simulation while collecting data from an RGB-D camera and the IMU sensor. From the RGB-D camera, we create a global Bird's Eye View (BEV) map containing height and color data at a 0.2 m resolution. From our IMU data, we take our z-axis acceleration and convert it into a normalized bandpower signal as a measure of traversability. Finally, we associate each patch with its bandpower signal and train a model to predict bandpower from the given patch.

A. Pre-filtering

Frames associated with high angular momentum (¿ 100) are filtered out because they tend to be associated with a glitch that occurs when hitting a tree and getting launched into the air.

Furthermore, cameras add a vignette effect to images, where the corners of images are darker than the middle. When present, this effect creates artifacts in the generated global maps. In order to remove this effect, the camera was calibrated using an image of a white floor.

B. Global Map

To create a global map from the raw camera data, we need to map each camera pixel into a global map pixel.

We start by transforming e ach pixel from a given depth image from camera coordinates to global coordinates. For this transformation we make use of the the absolute position, height, and orientation of the vehicle given by the simulator. We then use this transformation to map the RGB image pixels to the correct position on the global BEV map. We can also use the height of each transformed depth pixel to generate our height map.

Pixels that map to the same 0.2m square are averaged together. Additionally, we track the standard deviation of the height within each 0.2m square.

C. Patches

We take 5.4mx5.4m patches every 0.2 seconds along the trajectory. This covers the full footprint of the car plus some extra space. Patches are removed if they are more than 90% empty.

RGB data is converted into the HSV color space to be more robust to different lighting and shadows. Additionally, we use height data, 0.2mx0.2m height std data (capturing terrain roughness), and the 5.4mx5.4m height std data (capturing a larger scale slope). This gives our pathches a total of 6 channels.

D. Ground Truth Traversability

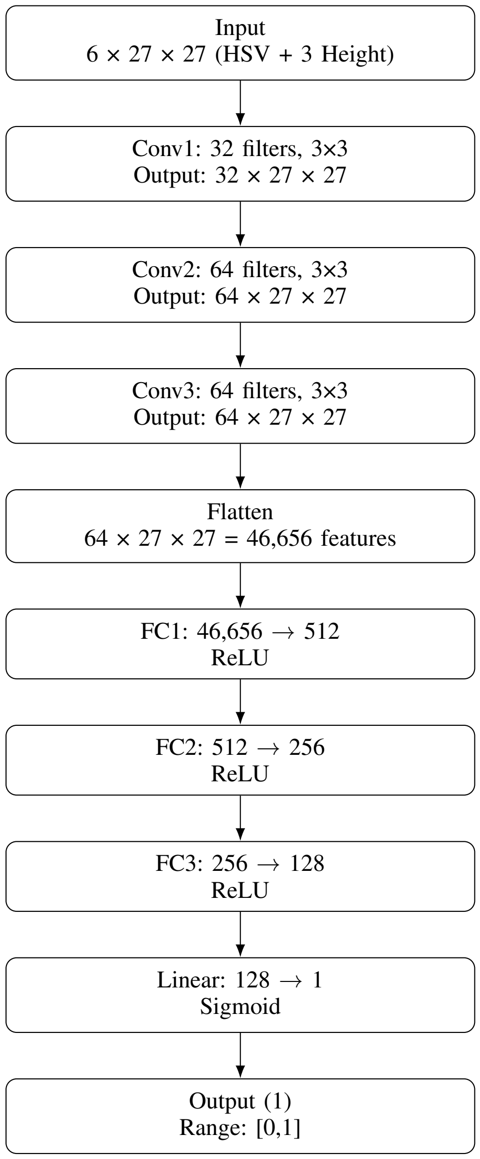
For our ground truth traversability we use the z-axis acceleration signal as used in [7].

...

We normalize bandpower to the range 0-1 and since bandpower generally has a few spots that are much greater than all the other spots, we set the whole top 5% of bandpower to 1.0.

E. CNN

We associate each patch with its sensed bandpower and train a CNN to predict the bandpower.



F. Experimental Setup

We use the CARLA simulator and partial Chrono physics, we are able to get relatively realistic visuals and decent physics. The main limitation of our simulation environment is that it does not support deformable terrain such as mud or sand.

We mount a virtual RGB-D camera on the roof of the car. Additionally, we allow access to the absolute position information of the car so that we do not have to perform Simultaneous Localization and Mapping (SLAM).

IV. Results

The model was trained on 66 runs over 2 different maps, CARLA Map 5 and CARLA Map 7, with a majority of data being taken on map 7. Overall this resulted in a total of 17,000 patches which were then randomly split into train, validation and test datasets while ensuring that all datasets have enough of the relatively rare high bandpower samples.

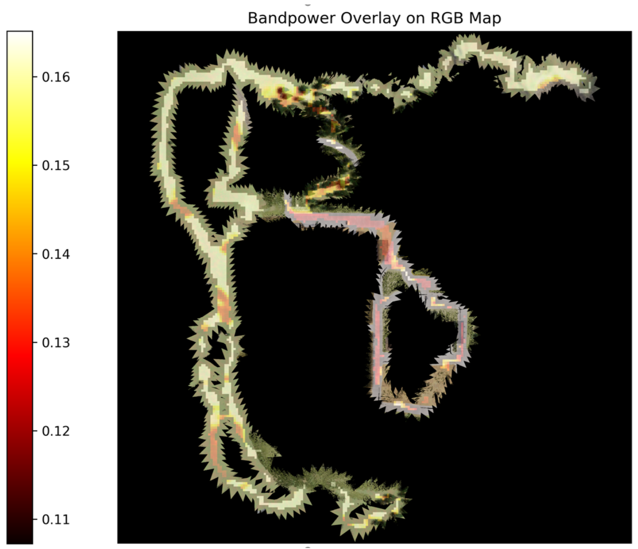


Fig. 1: Predicted bandpower on an unseen run on same map as training data. Notice that the road (red overlay) has the lowest predicted bandpower while the hills (white overlay) has the highest.

After training a model we can inspect the visual and numerical results. Looking at the visual results in Figure 1 we see that the model is making meaningful predictions. The road in the center of the map is predicted as having low bandpower while the hills near the edges of the map have a higher predicted bandpower. Meanwhile the trees have a predicted bandpower that is somewhere between that of the road and the hills.

Looking at the numerical results of training the model in Figure 2 we can see some of the hidden issues. The model predicts mostly around a value of 0.15 since the majority of samples are low. This allows it to achieve an MSE of 0.04 despite not matching the data very well. A more informative metric is the R^2 value which tells us the correlation between the models predictions and the true values. A value of 0 means no correlation while a value of 1 means fully correlated. We see that the R^2 value is 0.017 meaning there is only a very slight correlation.

V. Discussion

Overall, based on the visual results, we can say that the model was able to learn some meaningful information. However, the evaluation leaves something to be desired as the numerical results do not show that the model has learned anything at all. Future work will tackle this issue by comparing the average bandpower of segments to the average bandpower of the ground truth. These segments include roads, hills, trees, and cliffs. Another step that will be taken to improve the evaluation is to implement a planner that uses the generated traversability map to plan paths in the environment. We can then compare metrics of the path between different methods such as path bumpiness and task success rate as done in [3].

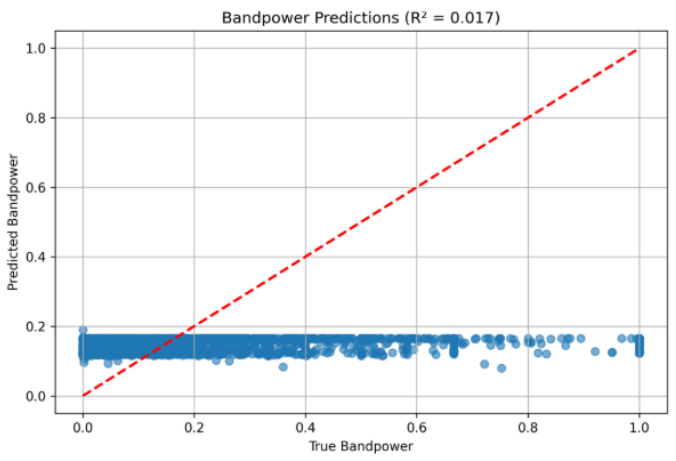


Fig. 2: Numerical results on the testing dataset. Notice how the true bandpower varies from 0 to 1 while the predicted bandpower is tightly clustered around 0.15. This is due to a highly unbalanced dataset.

References

[1] S. Pütz, T. Wiemann, J. Sprickerhof, and J. Hertzberg, "3D Navigation Mesh Generation for Path Planning in Uneven Terrain," in *Proc. 9th IFAC Symp. Intelligent Autonomous Vehicles (IAV)*, Leipzig, Germany, 2016, pp. 212–217.

[2] F. Schilling, X. Chen, J. Folkesson and P. Jensfelt, "Geometric and visual terrain classification for autonomous mobile navigation," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 2017, pp. 2678-2684, doi: 10.1109/IROS.2017.8206092.

[3] Zhang, B.; Chen, W.; Xu, C.; Qiu, J.; Chen, S. Autonomous Vehicles Traversability Mapping Fusing Semantic–Geometric in Off-Road Navigation. Drones 2024, 8, 496. https://doi.org/10.3390/ drones8090496

[4] S. Jung, J. Lee, X. Meng, B. Boots, and A. Lambert, "V-STRONG: Visual Self-Supervised Traversability Learning for Off-Road Navigation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Yokohama, Japan, 2024, pp. 8453–8460, doi: 10.1109/ICRA.2024.10610921.

[5] D. Kim, J. Sun, S. M. Oh, J. M. Rehg, and A. F. Bobick, "Traversability Classification Using Unsupervised On-Line Visual Learning for Outdoor Robot Navigation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Orlando, FL, USA, 2006, pp. 518–524, doi: 10.1109/ICRA.2006.1641796.

[6] F. Atas, G. Cielniak, and L. Grimstad, "From Simulation to Field: Learning Terrain Traversability for Real-World Deployment," *arXiv preprint arXiv:2501.06904*, 2025.

[7] M. G. Castro, S. Triest, W. Wang, J. M. Gregory, F. Sanchez, J. G. Rogers III, and S. Scherer, "How Does It Feel? Self-Supervised Costmap Learning for Off-Road Vehicle Traversability," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, London, U.K., 2023, pp. 931–938, doi: 10.1109/ICRA48891.2023.10161379.

[8] G. G. Waibel, P. Negrello, A. K. Tanwani, M. Chli, and R. Siegwart, "How Rough Is the Path? Terrain Traversability Estimation for Local and Global Path Planning," *IEEE Trans. Intelligent Transportation Systems*, vol. 23, no. 12, pp. 23917–23930, 2022, doi: 10.1109/TITS.2022.3150328.

[9] A. J. Sathyamoorthy, K. Weerakoon, T. Guan, J. Liang, and D. Manocha, "TerraPN: Unstructured Terrain Navigation Using Online Self-Supervised Learning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Kyoto, Japan, 2022, pp. 10557–10564, doi: 10.1109/IROS47612.2022.9982031.

[10] M. Bjelonic, S. Khattak, and M. Hutter, "Learning Multiobjective Rough Terrain Traversability," *Journal of Terramechanics*, vol. 101, pp. 11–25, 2022, doi: 10.1016/j.jterra.2021.12.004.

[11] J. Seo, T. Kim, S. Ahn, and K. Kwak, "METAVerse: Meta-Learning Traversability Cost Map for Off-Road Navigation," in *Proc. IEEE/RSJ*

Int. Conf. Intelligent Robots and Systems (IROS), Abu Dhabi, UAE, 2024, pp. 7768–7775, doi: 10.1109/IROS57659.2024.10801629.

[12] M. Sivaprakasam, P. Maheshwari, M. G. Castro, S. Triest, M. Nye, S. Willits, A. Saba, W. Wang, and S. Scherer, ”TartanDrive 2.0: More Modalities and Better Infrastructure to Further Self-Supervised Learning Research in Off-Road Driving Tasks,” *arXiv preprint arXiv:2402.01913*, 2024.