Real-time Semantic Mapping for Autonomous Off-Road Navigation

Daniel Maturana, Po-Wei Chou, Masashi Uenoyama and Sebastian Scherer

Abstract In this paper we describe a semantic mapping system for autonomous off-road driving with an All-Terrain Vehicle (ATVs). The system's goal is to provide a richer representation of the environment than a purely geometric map, allowing it to distinguish, e.g. tall grass from obstacles. The system builds a 2.5D grid map encoding both geometric (terrain height) and semantic information (navigation-relevant classes such as *trail*, *grass*, etc.). The geometric and semantic information are estimated online and in real-time from LiDAR and image sensor data, respectively. Using this semantic map, motion planners can create semantically aware trajectories. To achieve robust and efficient semantic segmentation, we design a custom Convolutional Neural Network (CNN) and train it with a novel dataset of labelled off-road imagery built for this purpose. We evaluate our semantic segmentation offline, showing comparable performance to the state of the art with slighly lower latency. We also show closed-loop field results with an autonomous ATV driving over challenging off-road terrain by using the semantic map in conjunction with a simple path planner. Our models and labelled dataset will be publicly available at http://dimatura.net/offroad.

1 Introduction

The last few years have seen enormous progress in the 3D sensing capabilities of autonomous vehicles. Mature and robust LiDAR and INS technologies give self-driving vehicles an accurate and real-time sense of the geometric structure around them, immensely simplifying navigation-related tasks.

Daniel Maturana · Po-Wei Chou · Sebastian Scherer
Robotics Institute, Carnegie Mellon University, e-mail: dimatura@cmu.edu

Masahi Uenoyama
Yamaha Motor Corporation USA, e-mail: mike_uenoyama@yamaha-motor.com

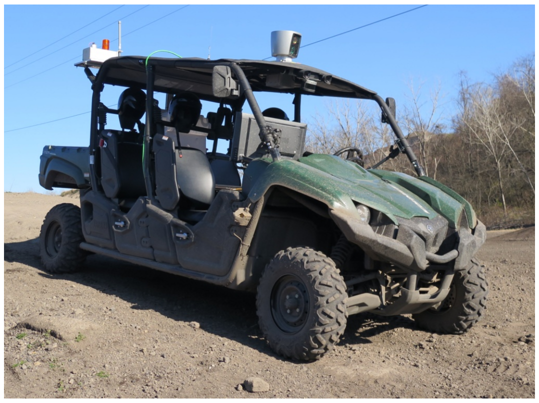


Fig. 1 Our Autonomous All-Terrain Vehicle (ATV). The two main sensors, a spinning 3D LiDAR and a stereo camera, can be seen on the vehicle roof.

However, we have observed that relying primarily on geometric information leads to disappointing results for autonomous navigation in off-road environments. The main reason is that geometric structure, by itself, fails to provide many important distinctions for wheeled All-Terrain Vehicles (ATVs) such as ours, shown in 1. For example, tall grass may be perceived as an obstacle, but our ATV may traverse it if desired. Similarly, leaf litter may appear as rocky terrain, or puddles may appear as either holes or smooth surfaces. All of these may lead to suboptimal, even dangerous, decisions in path planning. Similar observations have been made many times before in the context of off-road robotics, e.g., [12, 17, 24, 11].

In this paper, we describe a system to counter this problem by building a *semantic map*, a representation of the vehicle's surroundings encoding both geometric (e.g. height, roughness) and semantic information (navigation-relevant classes such as trail, grass, obstacle, etc.). The map is stored as a 2.5D grid centered on the vehicle frame and is continuously updated as new sensor data is acquired. Using this representation, a motion planner can create semantically-aware trajectories.

Our key contribution is a simple yet effective system coupling a custom Convolutional Neural Network architecture, based on Fully Convolutional Networks [16], and a 2.5D vehicle-centered semantic grid map that fuses the geometric and semantic measurements as the vehicle moves and acquires more data. We show the effectiveness of the semantic segmentation CNN in offline benchmarks. By using a simple planner with the semantic map, we show qualitative examples of our system being successfully used to navigate challenging off-road terrain.

As additional contributions, the labeled dataset of off-road imagery used to train our network, as well as our semantic segmentation code, will be made publicly available. See the project website, http://dimatura.net/offroad for links.

2 Related Work

Our system is heavily inspired by the rich literature on semantic approaches to off-road navigation tasks, going as far back as 1990 [7].

A decade later, various practical systems showed impressive results with this paradigm, usually with a combination of LiDAR and images [12, 17, 24, 27, 28].

The LAGR program [11] featured various highly relevant systems such as [20, 13, 26, 2], which performed semantic classification with hand-engineered vision pipelines. An exception is [9], featuring an early deep neural network system for semantic segmentation.

In more recent work, [23] demonstrate autonomous navigation featuring a lightweight semantic segmentation system. Unlike our system, they use traditional visual feature engineering, leading to noisy pixel-wise predictions that they smooth with a novel regularization method. In contrast, our architecture, based on Fully Convolutional Networks (FCNs) [16], incorporates spatial context that naturally smoothes the output. Another relevant work is [25], which uses an encoder-decoder network architecture that is similar to FCNs. They explore modalities beyond RGB and achieve impressive segmentation results. However, they do not build a metric map or demonstrate closed-loop navigation.

3 Approach

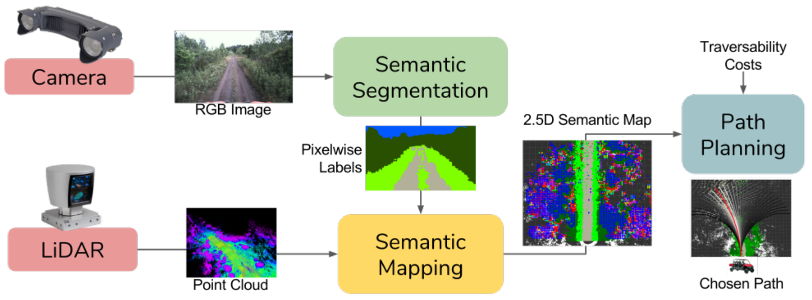


Fig. 2 Overview of our semantic mapping system.

Overview. This system architecture is outlined in Fig. 2. There are two primary sensor streams, RGB imagery and LiDAR point cloud data. The RGB images are fed into the semantic segmentation module, which uses a CNN to generate a pixel-wise labeling. Concurrently, the LiDAR point clouds are used to update a 2.5D grid map in the semantic mapping module. The semantic mapping module also receives the pixel-wise prediction images from the semantic segmentation module and projects them onto the 2.5D grid map, which fuses the semantic predictions over time. The result is a vehicle-centered 2.5D grid map encoding continuously updated estimates of relevant geometric and semantic information for off-road navigation. Finally, the map is used for semantically-aware path planning. For our initial testing, we used a simple receding horizon path planner that assigns a traversal cost to each semantic class and continuously chooses a path to minimize the cost. The whole system runs at 10 Hz, a rate dictated by the speed at which the semantic mapping module processes images.

Hardware Platform. Our vehicle is shown in Fig. 1. It is a commercial All-Terrain Vehicle modified and instrumented for experiments in autonomous off-road driving. The sensor suite includes an INS/GPS system, a 64-line Velodyne LiDAR and an RGB stereo camera with a 21 cm baseline manufactured by Carnegie Robotics. Note that the system in this paper does not currently use the stereo depth information. All computation is performed onboard with two COTS laptops, connected through high-speed ethernet. The laptop for semantic mapping includes an NVIDIA GT980M GPU, used to achieve real-time execution of the CNN classifier.

Software Platform. All computers run Ubuntu Linux. The different system modules run concurrently as ROS nodes and communicate through ROS messages. The nodes are implemented in C++ and Python, using CUDA (generated via the Theano library [3]) to make effective use of the GPU.

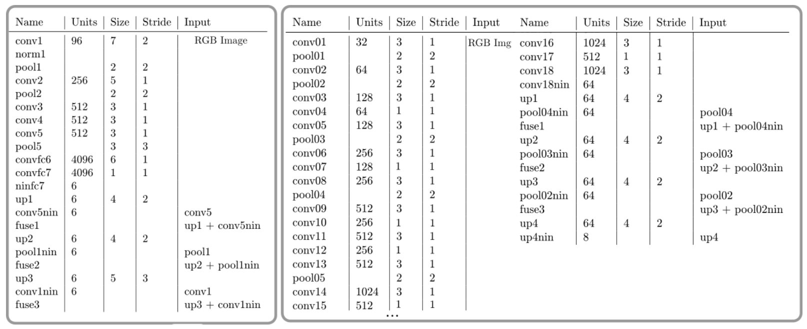
3.1 Semantic Segmentation

The goal of 2D semantic segmentation is to assign one of K predefined classes to each pixel in an image. Like many tasks in computer vision, the state of the art for this task has been recently revolutionized by Deep Learning, and in particular Convolutional Neural Networks.

For this task, the most successful neural networks architectures are Fully Convolutional Network (FCNs) [16]. The key idea in these networks is to take advantage of convolutional structure to label all the pixels simultaneously with a very similar network to more traditional CNNs. Due to pooling, this results in low-resolution outputs; to reverse this, so-called “deconvolution” layers are added to upsample the output. In order to preserve high-frequency detail, skip layers connecting early layers to upsampled feature maps are added. Encoder-Decoder architectures [19, 1], of which UpNet [25] is an example, are similar but omit skip layers.

At the start of the project, we found state of the art architectures to be relatively slow, as they were optimized for accuracy over speed. Thus we implemented and trained our own architectures, using the Theano [3] and Lasagne [6] libraries. We have found various possible architectures to show very similar accuracy for our off-road semantic segmentations tasks, differing mostly in time cost, which in turn is largely driven by details of the architecture and input/output resolution. We believe this is due to the relatively small datasets used.

We use two architectures. The first, `cnns-fcn`, is based on our "convolutionalization" of VGG-CNNs from [4], and has 227×227 input size with 109×109 output size. The second, `dark-fcn`, is based on our convolutionalization of the Darknet architecture [21], which in turn is similar but more efficient than VGG16 [22]. For `dark-fcn` both the input and output are 300×300, in order to facilitate comparison with UpNet. Despite the higher resolution `dark-fcn` is faster than `cnns-fcn`: 21 ms on a GT980M, compared to 37 ms. The authors of UpNet [25] describe a 50 ms with Caffe on a GTX Titan X, which in our experience has similar speeds to the GT980M. This leads us to believe our model should be faster or at least comparable. Fig. 3 shows both of our architectures. Code and trained models will also be made available.



Name	Units	Size	Stride	Input
conv1	96	7	2	RGB Image
norm1				
pool1		2	2	
conv2	256	5	1	
pool2		2	2	
conv3	512	3	1	
conv4	512	3	1	
conv5	512	3	1	
pool5		3	3	
convfc6	4096	6	1	
convfc7	4096	1	1	
ninfc7	6			
up1	6	4	2	
conv5nin	6			conv5
fuse1				up1 + conv5nin
up2	6	4	2	
pool1nin	6			pool1
fuse2				up2 + pool1nin
up3	6	5	3	
conv1nin	6			conv1
fuse3				up3 + conv1nin

`cnns-fcn`

Name	Units	Size	Stride	Input	Name	Units	Size	Stride	Input
conv01	32	3	1	RGB Img	conv16	1024	3	1	
pool01		2	2		conv17	512	1	1	
conv02	64	3	1		conv18	1024	3	1	
pool02		2	2		conv18nin	64			
conv03	128	3	1		up1	64	4	2	
conv04	64	1	1		pool04nin	64			pool04
conv05	128	3	1		fuse1				up1 + pool04nin
pool03		2	2		up2	64	4	2	
conv06	256	3	1		pool03nin	64			pool03
conv07	128	1	1		fuse2				up2 + pool03nin
conv08	256	3	1		up3	64	4	2	
pool04		2	2		pool02nin	64			pool02
conv09	512	3	1		fuse3				up3 + pool02nin
conv10	256	1	1		up4	64	4	2	
conv11	512	3	1		up4nin	8			up4
conv12	256	1	1						
conv13	512	3	1						
pool05		2	2						
conv14	1024	3	1						
conv15	512	1	1						
	...								

`dark-fcn`

Fig. 3 Our two network architectures. "conv" denotes a convolutional layer; "pool", a pooling layer; layers ending in "nin" are 1×1 convolutional layers; "fuse" is an elementwise sum layer; "up" is an upsampling deconvolution layer; "norm" is a local response normalization layer. The input is assumed to be the layer above, unless otherwise specified. For convolutional layers, "size" is the kernel size; for pooling layers, it is the pooling receptive field. Note that for `dark-fcn` we split the table due to space constraints.

3.2 Semantic Mapping

The output of the semantic mapping step is in 2D image space, but it is far more natural for vehicles to plan in a 3D, metric space. In our case, we adopt a 2.5D grid with each grid cell containing estimated height, i.e. a height map. This suffices for many environments, but would potentially have issues with overhanging trees or tunnels.

To keep an up-to-date height map of the vehicle's surroundings, we use a scrolling grid data structure, which has been reinvented multiple times in the literature. This structure is a generalization of ring-buffers to two dimensions, and its main feature is that it can be shifted (translated) without copying its data, and instead updating the variables indicating its limits. This is a speed optimization; logically, the grid behaves like a finite 2D array centered around the vehicle, with each grid cell containing various properties about the terrain. In our paper the grid cells are $0.25m \times 0.25m$ each, and the map has 400×400 cells. Each grid cell maintains a running estimate of the minimum and maximum height in that grid cell, computed by using occupancy and free-space constraints derived from LiDAR rays, similar to [8, 30]. For each point in the point cloud, we raytrace on our grid using Bresenham's algorithm in 3D; cells that are passed through, and above, are considered empty, and cells where the beam stops, and below, are considered occupied.

The semantic map also integrates semantic measurements, as its name indicates. To project the output of the 2D semantic segmentation into a height map representation, we follow a straightforward process depicted in Fig. 4.

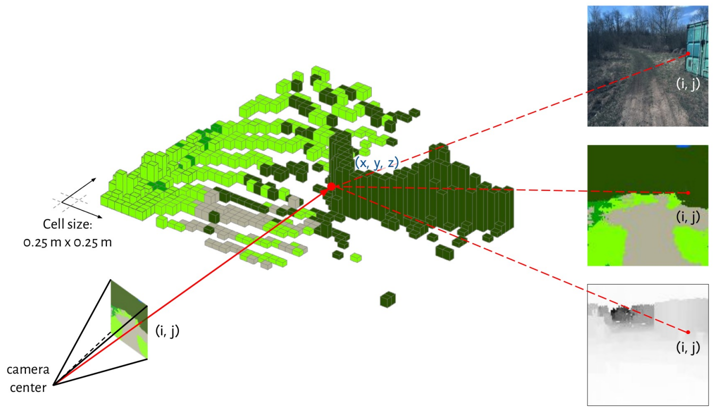


Fig. 4 Mapping the semantic segmentation to the 2.5D map.

Given that we know the relative position of the camera and the LiDAR, and the camera intrinsics, we can project the 2D semantic predictions onto the 2.5D grid cells using simple geometry. However, for added robustness, we fuse measurements over time. To this end we adopt a scheme inspired by the sequential filtering process of occupancy maps [18], but generalized to K classes.

For this, we use the probabilistic (softmax) pixel-wise output of the classifier. We maintain a running sum of the log odds of the K classes projected to each grid cell. While this soft multiclass representation could be used directly, for simplicity when interfacing with other systems, we use the argmax of the K classes as our current best estimate of the semantic class for each grid cell. Note that this representation assumes a single class per cell, which may be a limitation in certain environments.

An example cumulative output of the semantic map in a live field run is shown in Fig. 5.

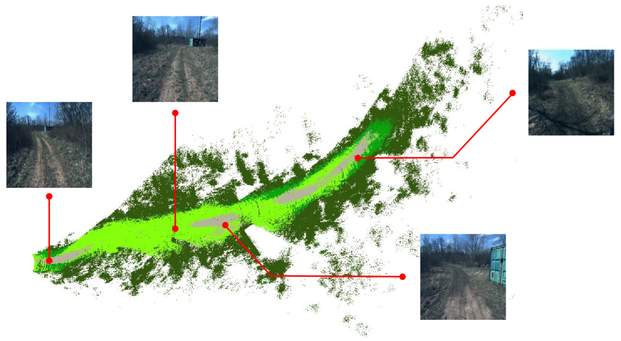


Fig. 5 Example output of semantic map in a field run.

3.3 *Path Planning*

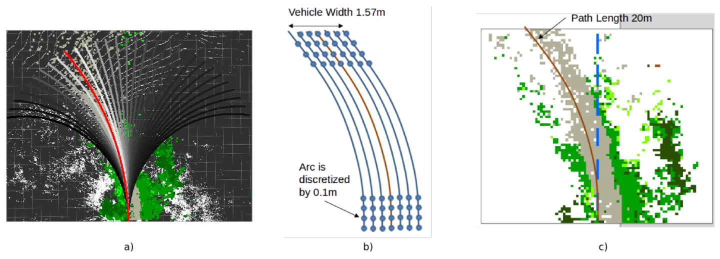


Fig. 6 Path Planning. a) Library of candidate paths, overlaid on top of the semantic map. Red indicates feasible paths. b) Illustration of how we account for vehicle width. For each trajectory, we compute the cost (or reward) over seven shifted versions of the trajectory, covering the vehicle footprint. c) An example of a chosen trajectory, chosen according to the traversability score of the semantic classes it covers.

In order to demonstrate autonomous operation, we implement an extremely simple receding horizon path planner. The planner has a library of 30 trajectories corresponding to yaw rates of $-15°/s$ to $15°/s$, discretized at $1°/s$, and at constant velocity of $9\,\mathrm{km\,h^{-1}}$; see Fig. 6a).

Each time the map is updated, which happens at 10 Hz, a trajectory is chosen from the library. The choice of trajectory maximizes a reward function derived from the semantic map as follows. Cells labeled as “smooth” or “rough” trail have a reward of 1, and cells labeled as “grass” have a reward of 0.1. All other classes have zero reward. The total reward of a trajectory is the sum of rewards over a 20 m trajectory length, originating from the vehicle. To account for vehicle width, we slightly modify this calculation, as shown in Figure 6b).

The advantage of this planner is that in its extreme simplicity, its performance depends largely on the output of our semantic mapping, with no interference from other factors that will be a present in a more complex, multi-layered system. However, our system was also used as an additional input to a more deliberative proprietary planner, for which the main representation was a geometric map built with LiDAR. In this planner, our semantic predictions were used primarily to avoid treating grass on and near trails as obstacles, enabling operation on narrow trails.

4 Experiments

Overview. We evaluate our system in two ways. First, we run offline benchmarks of the semantic segmentation module in two datasets. Second, we demonstrate the whole system operating autonomously live field experiments.

4.1 Offline Benchmarks

In order to evaluate our semantic segmentation module, we use two datasets, the DeepScene dataset from Valada et al. [25] and our own dataset, the Yamaha-CMU Off-Road Dataset.

DeepScene Dataset. This dataset consists of 233 training images and 139 validation images of off-road imagery densely labeled with six semantic categories: void, road, grass, vegetation, tree, sky, and obstacle. While this dataset shows some interesting variety in appearance due to the time of day, it is fairly small and seems to lack diversity in terms of weather and location. A key feature of this dataset is various modalities (depth, NIR), but we do not currently make use of them.

Yamaha-CMU Off-Road Dataset. In order to train and evaluate our method we have collected our own dataset, which we call Yamaha-CMU-Off-Road, or YCOR. It consists of 1076 images collected in four different locations in Western Pennsylvania and Ohio (8), spanning three different seasons (Fig. 7). The dataset was labeled using a polygon-based interface with eight classes: sky, rough trail, smooth trail, traversable grass, high vegetation, non-traversable low vegetation, obstacle. The polygon labels were post-processed using a Dense CRF [15] to densify the labels; the output of the CRF was manually inspected, and in some cases corrected, to ensure no wrong labels were created.

We believe our dataset is more diverse and challenging than DeepScene. In Fig. 8, we show the mean RGB image and pixel-wise label mode of each dataset. The DeepScene dataset shows a left-right bias and more predictable structure than ours; if we used the pixel-wise mode as a baseline classifier, we would obtain 0.30 pixel-wise error-rate in DeepScene, but 0.51 in ours. However, we acknowledge that compared to recent efforts, both datasets are relatively small; cf. CityScapes [5], with 25000 labeled images.

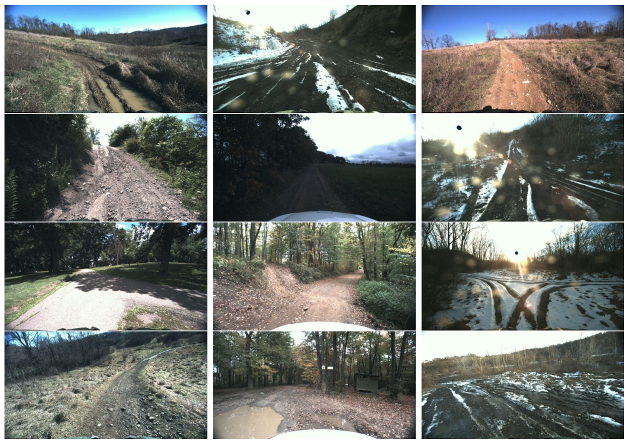


Fig. 7 Montage of frames from our dataset collection.

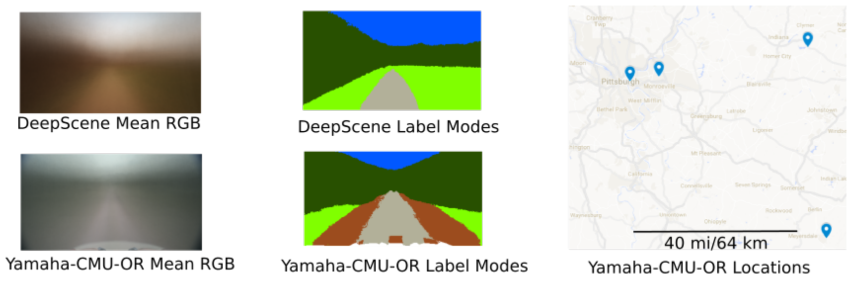


Fig. 8 First two columns: A comparison of dataset statistics. We show the mean RGB frame and the pixel-wise mode for the labeled frames in the training sets of each dataset used. Last column: a map with locations where YCOR was collected.

Our current split has 931 training images, and 145 validation images. This split was generated randomly, ensuring there was no overlap in data collection session between images in the training and validation split. However, there is overlap in locations used. We will provide location and time of acquisition metadata to enable further evaluation regarding generalization across these factors.

Quantitative Results. We evaluated our models on the two datasets. In each case, we train our models from scratch on the predefined training set until convergence with SGD, dividing by the initial learning rate (0.0001) by a factor of 10 three times. We use a standard pixel-wise cross-entropy loss with a small L_2 regularization factor (0.0005). Training takes around two days on a GT980Ti GPU. We use crop, rotation and color augmentations at training time, and none at test time. We use per-class intersection over union (IoU) as the evaluation metric, the most common metric for semantic segmentation.

Table 1 shows results for DeepScene and Table 2 shows results for YCOR. In both, we include a variant of the `dark-fcn` model with 448×448 resolution, in addition to the standard 300×300. We report the numbers from their paper [25], where we denote by frequency-weighted IoU (fw-IoU) what they denote as IoU, and add mean IoU (mIoU), calculated by ourselves. As we can see, both our models perform comparably, with `dark-fcn` having a slight advantage. In the DeepScene dataset we can also compare the two models with the RGB UpNet. We see that our models have a slight edge in fw-IoU, though they display dramatically worse performance for obstacles, which severely skews the mIoU metric. We note that the number of obstacle pixels in the dataset is three orders of magnitude less than for the other classes, so the network tends to ignore it. A similar situation occurs with puddles in YCOR. Nonetheless, it is an important class, and we are investigating how to detect it more accurately. Finally, we see that increasing the input resolution gives a slight boost in performance.

Table 1 Per-class, mean IoU and frequency-weighted IoU of UpNet (RGB) and our models in DeepScene dataset. The first three rows use a 300×300 image size, as in UpNet; the last row uses 448×448.

	road	grass	veg./tree	sky	obstacle	mIoU	fw-IoU
Upnet (RGB) [25]	85.03	**86.78**	**90.90**	90.39	**45.31**	**79.68**	85.30
`cnns-fcn`	85.95	85.34	87.38	90.53	1.84	58.51	87.47
`dark-fcn`	**88.03**	86.65	89.21	**93.17**	5.03	60.35	**89.41**
`dark-fcn-448`	88.80	87.41	89.46	93.35	4.61	60.61	89.85

Table 2 Per-class, mean IoU, and frequency-weighed IoU of our models in the YCOR dataset.

	smooth	grass	rough	puddle	obstacle	low veg.	high veg.	sky	mIoU	fw-IoU
`cnns-fcn`	**46.70**	64.03	38.29	0.0	32.74	24.32	79.15	88.01	46.66	61.31
`dark-fcn`	46.24	**71.25**	**41.35**	0.0	**29.74**	**28.17**	**80.15**	**91.45**	**48.54**	**63.62**
`dark-fcn-448`	52.46	72.11	39.61	0.0	35.56	24.61	82.51	92.69	49.82	65.18

Qualitative Results. We show some qualitative labelings of the `cnns-fcn` architecture for each dataset in Fig. 9. As can be seen, the results are generally accurate. For the YCOR, most of the confusions come from smooth vs. rough trail, a distinction that is hard for humans to make consistently.

4.2 Field Experiments

We performed various self-driving experiments in March and July 2017, in various locations around our testing site near Pittsburgh, PA. The terrain traversed including steep slopes, rocky and muddy terrain, puddles, and vegetation of various heights

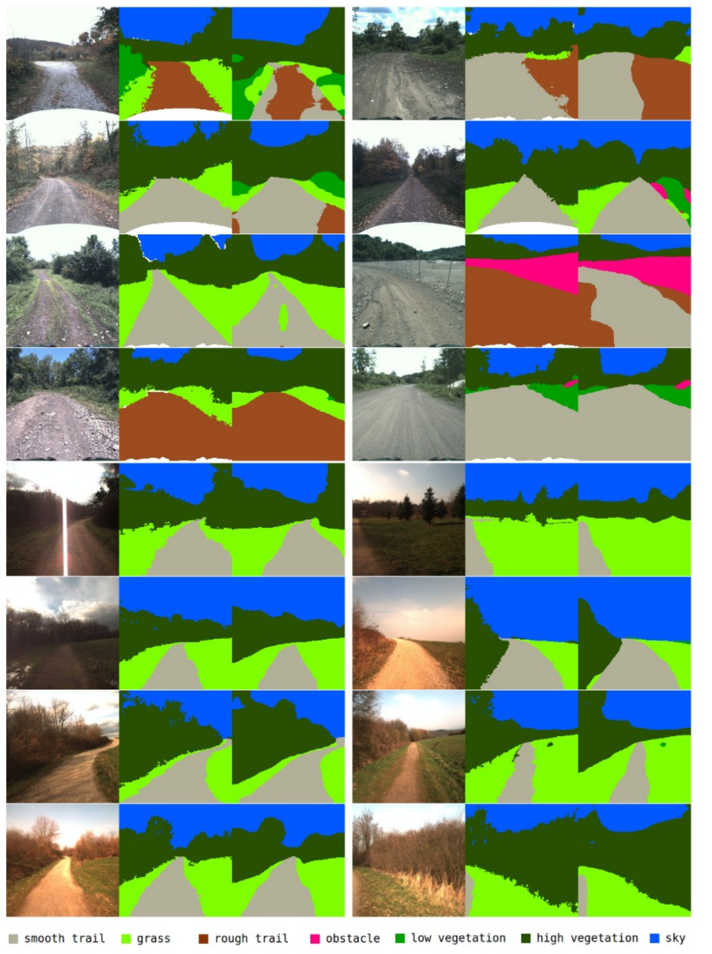


Fig. 9 Montage of predictions from `cnns-fcn` in the YCOR dataset (top four rows) and DeepScene (bottom four rows). In each case, we show three images: input, ground truth labels, and predicted labels.

surrounding and covering the trails. Despite the simplicity of our planner, the vehicle managed to successfully traverse various trails that were too challenging for a previous LiDAR-only system. These include locations with puddles, grass in the middle of the trail, and narrow trails. Video is available in the project website. Fig. 10 shows the vehicle in autonomous operation.

On the other hand, we observed some limitations of our current system. Many of the limitations were due to the simplicity of the receding horizon planner, which often swerved from side-to-side in wider trails.

Some of the failures were also due to our semantic classification system. For example, it sometimes failed to detect sparse grass alongside the trail, resulting in

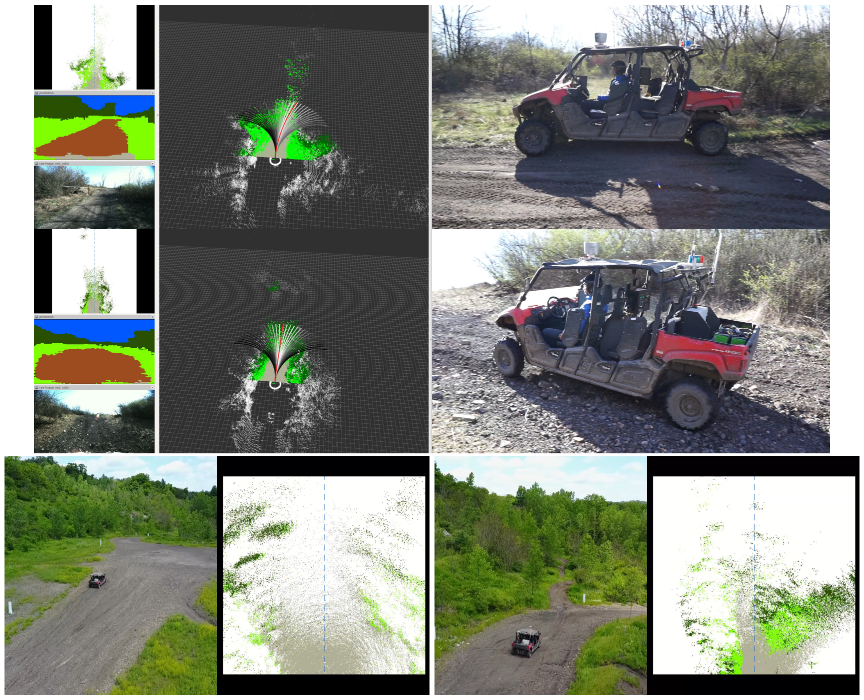


Fig. 10 Action shots of autonomous off-road driving in our testing site. In the first two rows, the left-hand side shows screenshots of the sensor data and map as seen from the vehicle. The last row shows aerial action shots with the right-hand showing the semantic map. Videos will be made available on the project page.

the vehicle veering off-trail. In one occasion, it also confused a large non-traversable bush with traversable grass, forcing us to manually intervene.

While we maintained a nominal speed of 9 km h^{-1}, the velocity in practice varied by a kilometers per hour; when traveling at more than 12 km h^{-1}, we occasionally observed the map would not update in time to make correct planning decisions, again resulting in failures to react appropriately.

More extensive testing was performed with the proprietary deliberative planner. In trials traversing more than 100 km, we observed far more stable operation.

Timing. We run all computation onboard the vehicle using an i7 laptop with a 6GB GT980M GPU. The bottleneck of the system is in the raytracing operation of semantic mapping, with semantic segmentation taking approximately 35 ms per image and the label projection taking around 60 ms per image. These steps occur sequentially, leading to the roughly 10Hz rate operation of the system. This is sufficient for medium-speed operation, but there is ample space to optimize performance futher to support faster driving and/or more limited computing platforms.

5 Conclusions

We have introduced an efficient and robust semantic mapping system for off-road navigation featuring a state-of-the-art CNN classifier. To train the CNN, we have collected and labelled a new dataset of off-road imagery. We have evaluated it in offline benchmarks with results comparable to the state of the art with lower latencies. We have also demonstrated closed-loop operation in challenging off-road terrain.

In future work, we are interested in incorporating recent advances from the state of the art in semantic segmentation, such as Dilation layers [29], pyramid spatial pooling [10]. We are also evaluating the contribution of multiple input modalities, including an approach jointly using LiDAR and imagery for segmentation [14].

Having verified firsthand the difficulty of accurately labelling large amounts of data, in future work we are interested in alternatives to manual labelling, such as self-supervision and inverse reinforcement learning.

Acknowledgements We gratefully acknowledge Yamaha Motor Corporation for their support, the Yamaha-CMU team for building and maintaining the autonomous ATV, and Mesh Robotics for field testing data.

References

1. Badrinarayanan, V., Handa, A., Cipolla, R.: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling (2015). URL http://arxiv.org/abs/1505.07293
2. Bajracharya, M., Howard, A., Matthies, L.H., Tang, B., Turmon, M.: Autonomous off-road navigation with end-to-end learning for the lagr program. Journal of Field Robotics **26**(1), 3–25 (2009)
3. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: SciPy (2010)
4. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. CoRR **abs/1405.3531** (2014). URL http://arxiv.org/abs/1405.3531
5. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
6. Dieleman, S., Schluter, J., Raffel, C., Olson, E., Sonderby, S.K., Nouri, D., et al.: Lasagne: First release. (2015). DOI 10.5281/zenodo.27878. URL http://dx.doi.org/10.5281/zenodo.27878
7. Fujimori, T., Kanade, T.: An approach to knowledge-based interpretation of outdoor natural color road scenes. In: Vision and Navigation: The Carnegie Mellon Navlab, pp. 39–81 (1990)
8. Hadsell, R., Bagnell, J.A., Huber, D., Hebert, M.: Non-Stationary Space-Carving Kernels for Accurate Rough Terrain Estimation. IJRR **29**(8), 981–996 (2010)
9. Hadsell, R., Erkan, A., Sermanet, P., Scoffier, M., Muller, U.: Deep belief net learning in a long-range vision system for autonomous off-road driving. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems **1**(1), 628–633 (2008)
10. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. CoRR **abs/1406.4729** (2014). URL http://arxiv.org/abs/1406.4729

11. Jackel, L.D., Krotkov, E., Perschbacher, M., Pippine, J., Sullivan, C.: The DARPA LAGR program: Goals, challenges, methodology, and phase i results. Journal of Field Robotics **23**(11-12), 945–973 (2006)
12. Kelly , A., Stentz , A.T., Amidi, O., Bode, M.W., Bradley , D., Diaz-Calderon, A., Happold, M., Herman , H., Mandelbaum, R., Pilarski, T., Rander, P., Thayer, S., Vallidis, N.M., Warner, R.: Toward reliable off road autonomous vehicles operating in challenging environments. The International Journal of Robotics Research **25**(5-6), 449–483 (2006)
13. Kim, D., Oh, S.M., Rehg, J.M.: Traversability classification for UGV navigation: A comparison of patch and superpixel representations. IEEE International Conference on Intelligent Robots and Systems pp. 3166–3173 (2007)
14. Kim, D.K., Maturana, D., Uenoyama, M., Scherer, S.: Season-Invariant Semantic Segmentation with A Deep Multimodal Network. In: FSR (2017)
15. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. CoRR **abs/1210.5644** (2012)
16. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
17. Manduchi, R., Castano, A., Talukder, A., Matthies, L.: Obstacle detection and terrain classification for autonomous off-road navigation. Autonomous robots **18**(1), 81–102 (2005)
18. Moravec, H., Elfes, A.: High resolution maps from wide angle sonar. In: International Conference on Robotics and Automation (ICRA) (1985)
19. Noh, H., Hong, S., Han, B.: Learning Deconvolution Network for Semantic Segmentation. CoRR **abs/1505.04366** (2015)
20. Procopio, M.J., Kegelmeyer, W.P., Grudic, G.: Terrain Segmentation with On-Line Mixtures of Experts for Autonomous Robot Navigation. Image (Rochester, N.Y.) pp. 385–397 (2009)
21. Redmon, J.: Darknet: Open source neural networks in c. http://pjreddie.com/darknet/ (2013–2016)
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)
23. Suleymanov, T., Paz, L.M., Pinis, P., Hester, G., Newman, P.: The path less taken: A fast variational approach for scene segmentation used for closed loop control. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3620–3626 (2016)
24. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., Mahoney, P.: Winning the DARPA grand challenge. Journal of Field Robotics (2006)
25. Valada, A., Olivera, G.L., Brox, T., Burgard, W.: Deep multispectral semantic scene understanding of forested environments. In: Proceedings of the 2016 International Symposium on Experimental Robotics (ISER) (2016)
26. Vernaza, P., Taskar, B., Lee, D.D.: Online, self-supervised terrain classification via discriminatively trained submodular Markov random fields. In: 2008 IEEE International Conference on Robotics and Automation, pp. 2750–2757. IEEE (2008)
27. Wellington, C., Courville, A., Stentz, A.: A Generative Model of Terrain for Autonomous Navigation in Vegetation. The International Journal of Robotics Research **25**(12), 1287–1304 (2006)
28. Wolf, D.F., Sukhatme, G.S.: Semantic mapping using mobile robots. IEEE Transactions on Robotics **24**(2), 245–258 (2008)
29. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. CoRR **abs/1511.07122** (2015)
30. Zhou, S., Xi, J., McDaniel, M.W., Nishihata, T., Salesses, P., Iagnemma, K.: Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain. Journal of Field Robotics **29**(2), 277–297 (2012)