Improving Traversability Estimation through Autonomous Robot Experimentation

Christos Sevastopoulos, Katerina Maria Oikonomou, and Stasinos Konstantopoulos

Institute of Informatics and Telecommunications,
NCSR 'Demokritos', Ag. Partaskevi, Greece
sev_chris@yahoo.com, k.m.oikonomou@gmail.com, konstant@iit.demokritos.gr

Abstract. The ability to have unmanned ground vehicles navigate unmapped off-road terrain has high impact potential in application areas ranging from supply and logistics, to search and rescue, to planetary exploration. To achieve this, robots must be able to estimate the traversability of the terrain they are facing, in order to be able to plan a safe path through rugged terrain. In the work described here, we pursue the idea of fine-tuning a generic visual recognition network to our task and to new environments, but without requiring any manually labelled data. Instead, we present an autonomous data collection method that allows the robot to derive ground truth labels by attempting to traverse a scene and using localisation to decide if the traversal was successful. We then present and experimentally evaluate two deep learning architectures that can be used to adapt a pre-trained network to a new environment. We prove that the networks successfully adapt to their new task and environment from a relatively small dataset.

Keywords: Robot vision · adaptability · self-assessment.

1 Introduction

The ability to have *unmanned ground vehicles (UGV)* navigate unmapped off-road terrain has high impact potential in application areas ranging from supply and logistics, to search and rescue, to planetary exploration. The core concept in outdoors navigation is the computation of a 2D *traversability grid* upon which path planning algorithms can operate. This grid is computed by using knowledge of the physical and mechanical properties and capabilities of a specific UGV in order to estimate if each grid cell can be safely traversed of not.

The two main lines of research in traversability estimate are *appearance-based* and *geometry-based* methods. The former rely on computer vision either directly estimate traversibility from raw images [1, 10] or to extract intermediate information that is then used in a separate traversability estimation step. Such information can be either terrain features such as roughness, slope, discontinuity and hardness [7] or terrain classes such as soil, grass, asphalt, vegetation [2]. *Geometry-based* methods, on the other hand, extract terrain features such as

roughness, slope, and discontinuity from *digital elevation maps (DEM)* obtained from stereoscopic or depth sensors [4, 13].

Methods that rely on DEM are more popular in recent literature as they estimate terrain features more directly and accurately than appearance-based methods. However, they lack the ability to distinguish between different classes of obstacles. For instance, a concrete obstacle and a patch of bushy vegetation can be practically impossible to distinguish in the DEM alone, although the distinction is important: a bush that can be easily overrun by the UGV should not be considered an obstacle. To address this, *hybrid methods* combine different types of sensory input to assess different aspects of traversability [5].

Hybrid methods, combined with recent advances in vision brought about by deep learning, can re-introduce appearance-based methods, and computer vision in general, as relevant to the outdoors traversability estimation problem. As demonstrated by recent work [1], a priorly learned network can be efficiently fine-tuned to a new environment using only a fraction of the data needed to initially train it. The fine-tuned network achieved an accuracy of 91% on unseen data, a marked improvement over the 75% achieved by the original network. However, in order to conduct their experiment Adhikari et al. had to manually annotate more than 5000 areas from 10 frames as 'traversible' or 'non-traversible'.

In the work described here, we also develop the idea of fine-tuning a generic network for a specific environment, but without requiring any manually labelled data. Instead, we present an autonomous data collection method that allows the robot to derive ground truth labels for scenes it observes and a network architecture that is fine-tuned from a relatively small, autonomously collected, collection of scenes. We prove the concept that the robot can autonomously assess the accuracy of its visual traversability estimation method, by attempting to traverse a scene and using localisation to decide if the traversal was successful (Section 2). We then discuss how the outcomes of this autonomous experimentation can be used to adapt the robot's visual traversability estimation models (Section 3), present experimental results that validate that where substantial adaptation is achieved from a feasible number of autonomous experiments (Section 4). We close the paper with conclusions and future reserach directions (Section 5).

2 Autonomous Data Collection

In order to have the robot autonomously fine-tune its traversability estimation, we want it to be able to measure the traversability of a path after traversing it (or having failed to traverse it). One key concept we are putting forward in this work is that traversability can be measured as the *error in proprioceptive localization*, that is, in localization that relies on the wheel encoder and IMU signals. The rationale is that along an easily traversible path, encoder drift is small and when fused with IMU becomes negligible. The more difficult a path is, the more the wheels drift and this error increases. In order to calculate this error, we will compare proprioceptive localization against the full 3D localization that fuses the encoders, IMU, and stereoscopic camera signals.

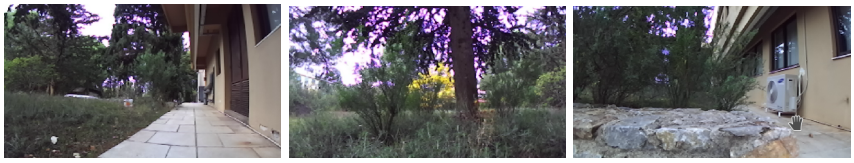

Fig. 1: Indicative scenes for the localization error experiments.

In order to prove this concept, we executed the following experiment using a DrRobot Jaguar rover fitted with a Zed stereoscopic camera: we selected several locations with varying degrees of traversability ranging from paved path, to vegetation that appears as an obstacle in the 3D point cloud created by the stereo camera but can be pushed back and traversed, to a wall (Figure 1 shows some indicative examples). After using standard ROS navigation to approach an obstacle, the robot synchronizes a secondary proprioceptive localization module with the main localization it uses to navigate, and then circumvents normal obstacle avoidance to push against the obstacle. The velocity is the minimum velocity that the robot can obtain, guaranteeing the safety of the platform even when pushing against a wall.

We have carried out 20 experiments, at varying locations and approaching the obstacle from different angles and have empirically found that for our platform and for our specific sensors a localization error of 21cm or less signifies that the path is traversible.[1] This simple rule achieves an accuracy of 9/10 in both the traversible and the non-traversible tests (Table 1). The specific threshold is likely to be different for different sensors and will need to be empirically identified for each robotic platform and mix of sensors, but after testing it on different kinds of obstacles we believe that it is a constant with respect to the environment.

We have looked at the outlying experiments and found that:

- The difference of 9cm on non-traversible path was caused by pushing against the wall from a very small angle, causing the platform to fall back to the paved road and continue moving.
- The difference of 89cm on a traversible path was caused by uneven slippage between the wheels on the one side and the wheels on the other side of the platform, as one side was on grass and the other on paved road. This does not happen when all wheels are on grass and slip evenly, as it is corrected by the IMU.

The former situation can be easily avoided by ensuring that the robot approaches the obstacle correctly. The latter situation is more difficult to detect and avoid from the localization and navugation system alone, and a future solution will involve reasoning over a more general situational awareness, so that the robot can determine if slippage is due to trying to push against an obstacle or due

[1] The software used and the data collected is publicly available at https://github.com/roboskel/traversability_estimation

to slippery terrain. For the experiments presented below we have included both false datapoints in the training set, to validate the robustness of the overall method against these labelling errors.

As we now have provided the robot with the means to use past experience as ground truth data, we will proceed in the next section to prove the concept that a reasonable amount of such data is sufficient to fine-tune visual traversability estimation for a new environment.

3 Adapting Traversability Estimation

In order to avoid requiring that the robot performs an unreasonable amount of experiments before observing any significant improvement in its traversability estimation capability, we assume as a starting point pre-trained deep learning model and experiment with how to use it as a generic feature extractor for classifiers trained on our environment-specific data.

We are going to use the VGG16 pre-trained model [11] as a feature extractor by exploiting its first five powerful convolutional blocks. In particular, its standard architecture comprises of a total of 13 convolutional layers using 3x3 convolution filters along with max pooling layers for down-sampling. In our case we will load the model with the Imagenet weights and freeze the convolutional blocks. As a result we take advantage of the output from the convolutional layers that represent high-level features of the data. By the process of flattening, this output is then converted to a one-dimensional vector that will be fed to the classifier. A commonly seen approach in literature involves adding one fully-connected (hidden) layer.

We have empirically found two hidden layers to give adequate accuracy while keeping the computational cost low, with 64 units for the first layer and 32 for the second layer. On both layers we use the 'relu' activation function [6]. Moreover, following Baldi and Sadowski [3], a dropout layer with a rate of 0.5 is inserted after each hidden layer. For the final output we add a softmax-activated layer with 2 units for the binary prediction; traversable or non-traversable. The reason we use the softmax activation function for the last layer is because of its ability to output the probability distribution over each possible class label [9, 11]

As for the adaptation itself, we have experimented with both *transfer learning* and *fine tuning*. In *transfer learning*, a base network is trained on an initial base dataset for the purposes of an initial task. Then, the learned features are

Table 1: The maximum difference (in cm) between full localization and proprioceptive localization, measured along the axis of the robot's movement. Using a threshold of 21cm, we get two mis-classifications in 22 runs.

Threshold:												21
Traversible:	0	1	1	2	6	7	7	8	14	16	18	89
Non-traversible:			110	30	28	28	27	26	25	25	22	9

transfered to a target network that is trained on a custom dataset for a different task. The technique we are going to implement is the combination of a *global average pooling layer* [11] one Dropout that tosses out 50% of the neurons and one dense layer that outputs the prediction of the two classes we are investigating. The reason for using the global average tecnhique is that due to its structural regularization nature, it offers an alternative approach in preventing overfitting of the fully connected layers.

In *fine tuning*, the process we followed is to freeze the weights of the low level convolutional layers and then add dense layers on top. We expect the model to be able to map the knowledge represented in the low level convolutional layers to the desired environment-specific output.

4 Experimental Results and Discussion

In order to validate our approach, we will use data collected following the method presented in Section 2 in order to adapt the pre-trained VGG16 object recognition network to our task *and* our environment using the two methods presented in Section 3. That is, we aim at adapting not only to improve for a new environment, but also at tranferring VGG16 knowledge to the new classification task (traversible vs. non-traversible) for which the robot is able to autonomously collect labelled data. We then evaluate the performance of these two two adapted networks against a baseline that makes traversability decisions based on the object recognition results from the original VGG16.

4.1 Data Acquisition

Our dataset comprises the RGB frames from the experiments described in Section 2. We have randomly split the runs between training, validation, and testing, but forcing the two mis-classified runs to be placed in the training set. This was done in order to (a) estimate the robustness of the approach to the errors made by our autonomous labelling method; and (b) to ensure that testing set us perfectly labelled in order to get accurate evaluation results. The resulting dataset consists of 1300 training images, 150 validation images, and 150 testing images.

It should be noted that the split was done at the level of individual runs. In this manner, although all data is from the NCSR 'Demokritos' campus, the testing data is from *different* locations within this general environment than the training and validation data.

4.2 Baseline

As a baseline, we will assume inferring traversability from the objects recognized by the unmodified, pre-trained VGG16 network. We started by manually mapping all VGG classes that appear in our dataset to 'traversible' (e.g., vegetation) or 'non-traversible' (e.g., stonewall, vehicle). We then assume the three most-probably classes, as probabilities after the third prediction become negligible ($\sim 10^{-4}$). We then apply the following rules:

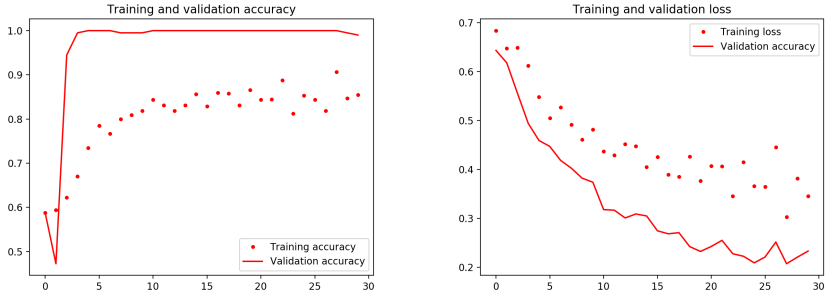

Fig. 2: Accuracy and loss for Model 1

- If all predictions are positive (traversible), the scene is assumed to be positive.
- If the most probable prediction is positive with probability p_{pos}, and the most likely negative class has a probability p_{neg}, then the scene is assumed to be positive if $p_{pos} - p_{neg} > 0.01$ and negative otherwise.
- If the most probable prediction is negative with probability p_{neg}, and the other two predictions are both positive and are both considerably likely, the scene is assumed to be positive. Specifically, if the two positive recognitions have probabilities p^a_{pos} and p^b_{pos}, then if must be that $p_{neg} - p^a_{pos} < 0.2$ and $p_{neg} - p^b_{pos} < 0.5$.
- Otherwise, the scene is assumed to be negative (non-traversable).

We have designed these rules so that they push the balance towards positive predictions. We have also experimented with more straightforward inferences, such as simply assuming the traversability label of the single most probable object, but we have observed that such rules tend to over-generalize in favour of negative predictions. We have emprically found these rules and thresholds to give the best possible results for our dataset, setting the baseline par as high as possible when using the pre-trained VGG16 network.

Qualitatively speaking, observing VGG16 predictions we found some completely erroneous recognitions, as well as some instances where correctly recognized non-traversible objects should not have been the dominant characterization of the scene but a minor one. But in general, most predictions were correct *and* meaningful for our task.

4.3 Training

We utilized the Keras deep-learning framework. First, we resized all images to 224x224, which is the original ImageNet format. We also apply *data augmentation* to increase dataset size by rotating, shifting and zooming on the initially training dataset.

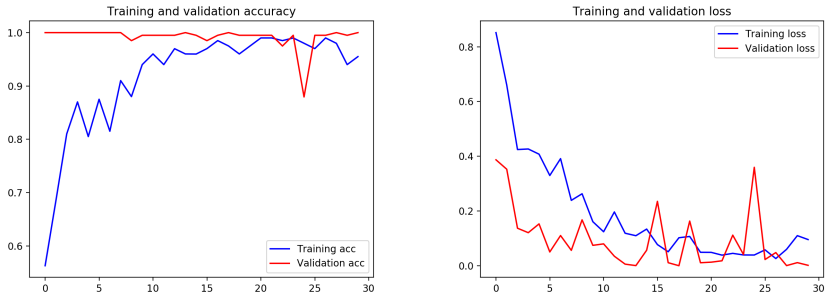

Fig. 3: Accuracy and loss for Model 2

We then built two models with two different front layer architectures. For Model 1 we used the RMSProp optimizer [12] and a learning rate of 10^{-3}; for Model 2 we used the *Adam* optimizer [8] and a learning rate of 10^{-4}. These learning rates were estimated by starting at a value of 10^{-2} and decreasing exponentially. Also, since our classification goal is binary, we are using the *binary_crossentropy* loss function.

Training Model 1 for 30 epochs, we get the accuracy and loss transfer shown in Figure 2. We can see that the model performs well on both the the training and validation sets. More papecifically, the Loss plots for both training and validation constantly decrease and stabilize around the same point. As a consequence, we can deduce that the model is learning and generalizing in an adequate manner.

Training Model**??** 2 for 30 epochs, we get the the accuracy and loss given in Figure 3. Overall, the model's performance is satisfactory. However, we can observe certain signs of fluctuation, revealing that potentially there has not been adequate generalization.

4.4 Results

In order to infer a binary traversability decision, comparable to the decision made by our raw-VGG baseline system, we assume as traversible the scenes where the 'traversible' probability is at least 0.075 higher than the 'non-traversible' probability. Table 2 gives the accuracy of the baseline and our two models per traversability category.

It can be noticed that, as already discussed above, VGG errs on the side of safety and its retrieval is low on traversable examples. Our custom models also show higher confidence (Figure 4), which is to be expected as they have been tuned to our (simpler, binary) task.

4.5 Error analysis

We observe that both the performances of Model 1 and Model 2 outperform VGG when we are investigating a traversable scene. One common failure, however, is

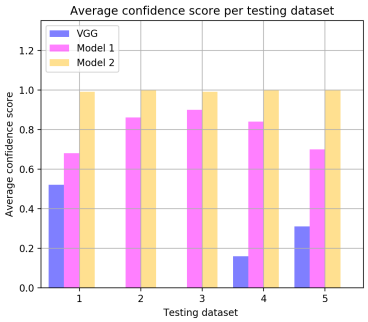

Fig. 4: Average Conficence scores per testing dataset

(a) Traversable vegetation (b) Non-traversable (c) Stone wall

Fig. 6: Some characteristic instances

caused by the presence of an non-traversable objects, such as the building on the left of Figure 6a. Although this is a traversable scene, our methods does not focus on the part of the scene directly in front of the robot, but characterizes the scene as a whole.

What is more, there are examples that although the first output of VGG corresponds to a class that has been defined as 'traversable', secondary recognitions lead to mis-classifications. For instance, the scene in Figure 6b is classified as 'lakeside' and 'park bench', with a small probability difference between the two.

Finally, an example where the baseline is outperformed is given in Figure 6c. This scene is recognized as 'patio' and 'stone wall', the latter at an exceptionally low probability leading the baseline to classify this as a traversable scene. Model 1

System	Accuracy	
	Traversible examples	Non-traversible examples
VGG	33.3	97.3
Model 1	99.1	100
Model 2	98.2	100

Table 2: Accuracy of the baseline and our two models on traversible and non-traversible training data.

and Model 2 considered the presence of the stone wall and correctly classified the scene as non-traversable.

5 Conclusion and Future Work

We presented a method that tackles the problem of adapting generic vision problems to a new task and environment by only using whatever labelled data the robot is able to collect autonomously and without any human supervision. The core of the learning method is to build CNNs that can take advantage of *transfer learning* and *fine tuning* to keep the convolutional architecture of the VGG intact.

Overall despite the small number of training data, we evaluated our models on the testing dataset and we witnessed a significant improvement over the VGG's weak performance, especially on traversable scenes. Regarding non-traversable scenes where the predictive capabilites of VGG are indisputable, we were able to take advantage of its convolutional features and transfer its obstacle recognition ability to our models. Hence, we showed that the robot can make predictions about its environment by using a reasonable amount of data, and in fact data that is labelled without any manual annotation.

A common problem that our approach cannot completely address is the lack of a sense of specific traversible paths through a generally non-traversible scene, making the overall methodology unnecessarily cautious. In future work we plan to extend the method so that it estimates the traversability of specific paths rather than of the scene as a whole. One approach we will experiment with will be cropping the test images so that traversability is estimated on a narrower field around the selected path; and, accordingly, collecting training data from similarly cropped images around the path tested during autonomous experimentation.

References

[1] Adhikari, S.P., Yang, C., Slot, K., Kim, H.: Accurate natural trail detection using a combination of a deep neural network and dynamic programming. Sensors 18(1) (Jan 2018), https://doi.org/10.3390/s18010178

[2] Angelova, A., Matthies, L., Helmick, D., Perona, P.: Fast terrain classification using variable-length representation for autonomous navigation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007). pp. 1–8. IEEE (2007)

[3] Baldi, P., Sadowski, P.J.: Understanding dropout. In: Proceedings of the 2013 Conference on Neural Information Processing Systems (NIPS 2013) (2013)

[4] Bellone, M.: Watch your step! terrain traversability for robot control. In: Gorrostieta Hurtado, E. (ed.) Robot Control, chap. 6. InTech (Oct 2016)

[5] Bellutta, P., Manduchi, R., Matthies, L., Owens, K., Rankin, A.: Terrain perception for demo iii. In: Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE. pp. 326–331. IEEE (2000)

[6] Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 315–323 (2011)
[7] Howard, A., Seraji, H., Tunstel, E.: A rule-based fuzzy traversability index for mobile robot navigation. In: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on. vol. 3, pp. 3067–3071. IEEE (2001)
[8] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of ICLR 2015 (2015), arXiv:1412.6980
[9] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
[10] Rasmussen, C., Lu, Y., Kocamaz, M.: Appearance contrast for fast, robust trail-following. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009), St. Louis, MO, USA, 10–15 October 2009 (2009)
[11] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proceedings of the 2015 International Conference on Learning Representations (2015), arXiv:1409.1556
[12] Tieleman, T., Hinton, G.: RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning 4(2) (2012)
[13] Wermelinger, M., Fankhauser, P., Diethelm, R., Krüsi, P., Siegwart, R., Hutter, M.: Navigation planning for legged robots in challenging terrain. In: Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016), Daejeon, South Korea, October 2016 (Oct 2016)