Traversability Estimator for Legged robot.

Subtitle: bho

Master's Thesis submitted to the
Faculty of Informatics of the *Università della Svizzera Italiana*
in partial fulfillment of the requirements for the degree of
Master of Science in Informatics
Ask

presented by
Francesco Saverio Zuppichini

under the supervision of
Prof. Student's Alessandro Giusti
co-supervised by
Prof. Student's Co-Advisor

June 2019

Contents

Contents **A**
0.1 Introduction C
0.2 Related Work . . . . . D
0.3 Results . . . . . F
0.3.1 Experiment Setup . . . . . F
0.3.2 Quantitative Results . . . . . G
0.3.3 Final results . . . . . G
0.3.4 Qualitative results . . . . . H
0.4 Interpretability . . . . . J

Abstract

With this project, we estimate ground traversability for a legged crocodile-like robot. We generate different synthetic grounds and let the robot walk on them in a simulated environment to collect its interaction with the terrain.

Then, we train a deep convolutional neural network using the data collected through simulation to predict wether a given ground patch can be traverse or not. Later, we highlight the strength and weakness of our method by using interpretability techniques to visualise the network's behaviour in interesting scenarios.

0.1 Introduction

Effective identification of traversable terrain is essential to operate mobile robots in every type of environment. Today, there two main different approaches used in robotics to properly navigate a robot: online and offline. The first one uses local sensors to map the surroundings "on the go" while the second equip the mobile robot with an already labeled map of the terrain.

In most indoor scenarios, specific hardware such as infrared or lidar sensors is used to perform online mapping while the robot is exploring, this is the case of the most recent vacuum cleaner able to map all the rooms in an apartment. With the recent breakthroughs of deep learning in computer vision, more and more cameras have been used in robotics. For example, self-driving cars utilize different cameras around the vehicle to avoid obstacle using object detection. Indoor scenarios share similar features across different places shifting the problem from which ground can be traversable to which obstacle must be avoided. For instance, the floor is always flat in almost all rooms due to is artificial design. Moreover, usually, traversability must be estimated on the fly due to the high number of possible obstacles and to the layout of the objects in each room may not be persistent in time.

On the other hand, outdoor scenarios may have less artificial obstacle but their homogeneous ground makes challenging to determine where the robot can properly travel. Moreover, a given portion of the ground may not be traversable by all direction due to the not uneven terrain. Fortunately, a height map of the ground can be obtained easily by using third-party services or special flying drones. Those maps are extremely valuable in robotics applications since they provide an efficient way to examine the features of terrains, such as bumps, holes, and walls.

These scenarios have different difficulties. In indoor environments, it is easier to move the robot on the ground since it is designed for humans, but harder to perform obstacle avoidance. While in outdoors scenario it is maybe more challenged to the first estimate where the robot can go due to the huge variety of ground features that may influence

traversability. To learn where to move, an artificial controller must be trained to predict the robot interaction with the environment. Such a process requires to collect some data to train the controller. However, this may not be a straight forward process. In indoors terrain, most of the times, data is collected by driving the robot directly in the environment by a human or an artificial controller. While in the outdoors scenario data is assembled using the simulation for convenience.

Our approach aims to estimate traversability of a legged robot krocodile-like robot called Krock. We generate different uneven grounds in form of height map and let the robot walk for a certain amount of time on each one of them while recording its interaction, position and orientation. After, for each stored robot position we crop the corresponding ground portion in which the robot was during the simulator, those patches composed the training dataset. We select a minimum space in a fixed amount of time that the robot must travel to successfully traverse a ground region and use it to label the dataset. Then, we fit a deep convolutional neural network to predict the traversability probability. Later, we evaluate it using different metrics using real-world terrains.

The report is organized as follow, the next chapter introduces the related work, Chapter 2 describes our approach, Chapter 3 talks in deep about the implementation details, Chapter 4 shows the results and Chapter 5 discuss conclusion and future work.

0.2 Related Work

The learning and perception of traversability is a fundamental competence for both organisms and autonomous mobile robots since most of their actions depend on their mobility [12]. Visual perception is known to be used in most all animals to correctly estimate if an environment can be traversed or not. Similar, a wide array of autonomous robots adopt local sensors to mimic the visual properties of animals to extract geometric information of the surrounding and plan a safe path through it.

Different methodologies have been proposed to collect the data and then learn to correctly navigate the environment. Most of the methodologies rely on supervised learning, where first the data is gathered and then a machine learning algorithm is trained sample to correctly predict the traversability of those samples. Among the huge numbers of methods proposed, there are two categories based on the input data: geometric and appearance based methods.

Geometric methods aim to detect traversability using geometric properties of surfaces such as distances in space and shapes. Those properties are usually slopes, bumps, and ramps. Since nearly the entire world has been surveyed at 1 m accuracy [10], outdoor robot navigation can benefit from the availability of overhead imagery. For this reason, elevation data has also been used to extract geometric information. Chavez-Garcia et al. [2], proposed a framework to estimate traversability using only elevation data in the form of height maps.

Elevation data can also be estimated by flying drones. Delmerico et al. [5] proposed a collaborative search and rescue system in which a flying robot that explores the map and creates an elevation map to guide the ground robot to the goal. They train on the fly a convolutional neural network to segment the terrain in different traversable classes.

Whereas appearance methods, to a greater extent related to camera images processing and cognitive analyses, have the objective of recognising colours and patterns not related

to the common appearance of terrains, such as grass, rocks or vegetation. Those images can be used to directly estimate the traversability cost.

Historically, the collected data is first preprocessed to extract texture features that are used to fit a classic machine learning classified such us an SVM [12] or Gaussian models [10]. Those techniques rely on texture descriptors, for example, Local Binary Pattern [11], to extract features from the raw data images obtained from local sensors such as cameras. With the rise of deep learning methods in computer vision, deep convolution neural network has been trained directly on the raw RGB images bypassing the need to define characteristic features.

One recent example is the work of Giusti et al. [1] where a deep neural network was training on real-world hiking data collected using head-mounted cameras to teach a flying drone to follow a trail in the forest. Geometric and appearance methods can also be used together to train a traversability classifier. Delmerico et al.[3] extended the previous work [5] by proposing the first on-the-spot training method that uses a flying drone to gather the data and train an estimator in less than 60 seconds.

Data can also extract in simulations, where an agent interacts in an artificial environment. Usually, no-wheel legged robot able to traverse harder ground, can benefits from data gathering in simulations due to the high availability. For example, Klamt et al. [6] proposed a locomotion planning that is learned in a simulation environment.

On other major distiction we can made is between different types of robots: wheel and no-wheel. We will focus on the later since we adopt a legged crocodile-like robot to extend the existing framework proposed by Chavez-Garcia et al. [2].

Legged robots show their full potential in rough and unstructured terrain, where they can use a superior move set compared to wheel robots. Different frameworks have been proposed to compute safe and efficient paths for legged robots. Wermelinger et al. [**wermelinger2016navigation**] uses typical map characteristics such as slopes and roughness gather using onboard sensors to train a planner. The planner uses a RRT* algorithm to compute the correct path for the robot on the fly. Moreover, the algorithm is able to first find an easy local solution and then update its path to take into account more difficult scenarios as new environment data is collected.

Due to uneven shape rough terrain, legged robots must be able to correctly sense the ground to properly find a correct path to the goal. Wagner et al. [7] developed a method to estimate the contact surface normal for each foot of a legged robot relying solely on measurements from the joint torques and from a force sensor located at the foot. This sensor at the end of a leg optically determines its deformation to compute the force applied to the sensor. They combine those sensors measurement in an Extended Kalman Filter (EKF). They showed that the resulting method is capable of accurately estimating the foot contact force only using local sensing.

While the previous methods rely on handcrafted map's features extraction methods to estimate the cost of a given patch using a specific function, new frameworks that automatise the features extraction process has been proposed recently. Lorenz et al. [**wellhausen2019where**] use local sensing to train a deep convolutional neural network to predict terrain's properties. They collect data from robot ground interaction to label each image in front of the robot in order to predict the future interactions with the terrain showing that the network is perfectly able to learn the correct features for different terrains. Furthermore, they also perform weakly supervised semantic segmentation using the same approach to divide the input images into different ground classes, such as glass and sand,

showing respectable results.

Add mirko's paper

0.3 Results

In the section we show and evaluate the model's results. We will start by presenting to the reader the networks score on each metric, then we will use the best model to predict the traversability in real world terrains.

0.3.1 Experiment Setup

Hardware

We run all the experiment on a Ubuntu 18.10 work station equipped with a Ryzen 2700x, a powerful CPU with 8 cores and 16 threads, and a NVIDIA 1080 GPU with 8GB of dedicated RAM.

Dataset

To perform classification, we select a threshold of 0.2m on a time window, Δt, of two seconds to label the patches, meaning that a patch with an advancement less than 20 centimeters is labeled as *no traversable* and viceversa. The This processed is explained in detail in the previous chapter. While for the regression, we did not label the patch and directly regress on the advancement.

Initially, to train the models we first use Standard Gradient Descent with momentum set to 0.95 and weight decay to $1e-4$ with an initial learning rate of $1e-3$ as was originally proposed to train residual network [4]. However, we later utilize Leslie Smith's 1cycle policy [9] that allows us to trian the network faster and with an higher accuracy. We minimise the binary Cross Entropy for the classifier and the Mean Square Error (MSE) for the regressor.

Experimental validation

We select as *validation* set ten percent of the training data. Since we store each run of Krock as a *.csv* file, validation and train set do not overlap. The test set is composed entirely by the Quarry map, a real world scenario. Table **??** tells in detail the configuration used in each map of all sets.

We also evaluate the model on the following additonal maps

add arck rocks

Metrics

Classification: To evaluate the model's classification performance we used two metrics: *accuracy* and *AUC-ROC Curve*. Accuracy scores the number of correct predictions made by the network while AUC-ROC Curve represents degree or measure of separability, informally it tells how much model is capable of distinguishing between classes. For each experiment, we select the model with the higher AUC-ROC Curve during training to be evaluated on the test set.

Regression: We used the Mean Square Error to evaluate the model's performance.

0.3.2 Quantitative Results

Model selection

We compared two different *micro-resnet* and the *vanilla* cnn from the previous Chapter. We evaluate those models using a time window of two second, a threshold of 20cm and the data augmentation techniques described before. We run five experiments per architecture and we select the best performing network, the results are showed in the following table.

		Vanilla	MicroResnetSE	
			3×3 stride 1	7×7 stride 2
AUC	Top	0.892	0.888	**0.896**
	Mean	**0.890**	0.883	0.888
Params		974,351	313,642	314,282

Table 1. Model comparison on the test set.

Luca told me is better to split the models like Model1 and Model2 etc

Based on this data We select *micro-resnet* with squeeze and excitation and a starting convolution's kernel size of 7×7 with stride of 2. This model has one third of the parameter of the origal model proposed by Chavez-Garcia et all [2].

As proof of work, we also train the best network architecture, MicroResnetSE with a first convolution's kernel size of 7×7 and stride= 2, with and without the Squeeze and Excitation operator.

	MicroResnet7×7	MicroResnet7×7-SE	Improvement
Top	0.875	**0.896**	+0.021
Mean	0.867	**0.888**	+0.021

Table 2. AUC top value and mean value for MicroResnet with a fist convolution of 7×7 and stride = 2 with and without the SE module. The improvement is the same.

0.3.3 Final results

The following table shows in deep the score of the best network for each dataset.

I have actually never talk about surf rocks

Moreover, we would like to also show the different steps we made to reach this result. The following table shows the metric's score without any data-augmentation.

add result with and without data agu

Adding dropout increases the results.

table with results

Dataset			micro-resnet		Size	Resolution(cm/px)
Type	Name	Samples	ACC	AUC		
Synthetic	Training	429312	-	-		2
	Validation	44032	95.2 %	0.961		2
	Arc Rocks	37273	85.5 %	0.888		2
Real evaluation	Quarry	36224	88.2 %	0.896		2
	foo	TODO				
	baaa	TODO				

Table 3. Final results on different datasets.

With dropout plus coarse dropout.

table with results

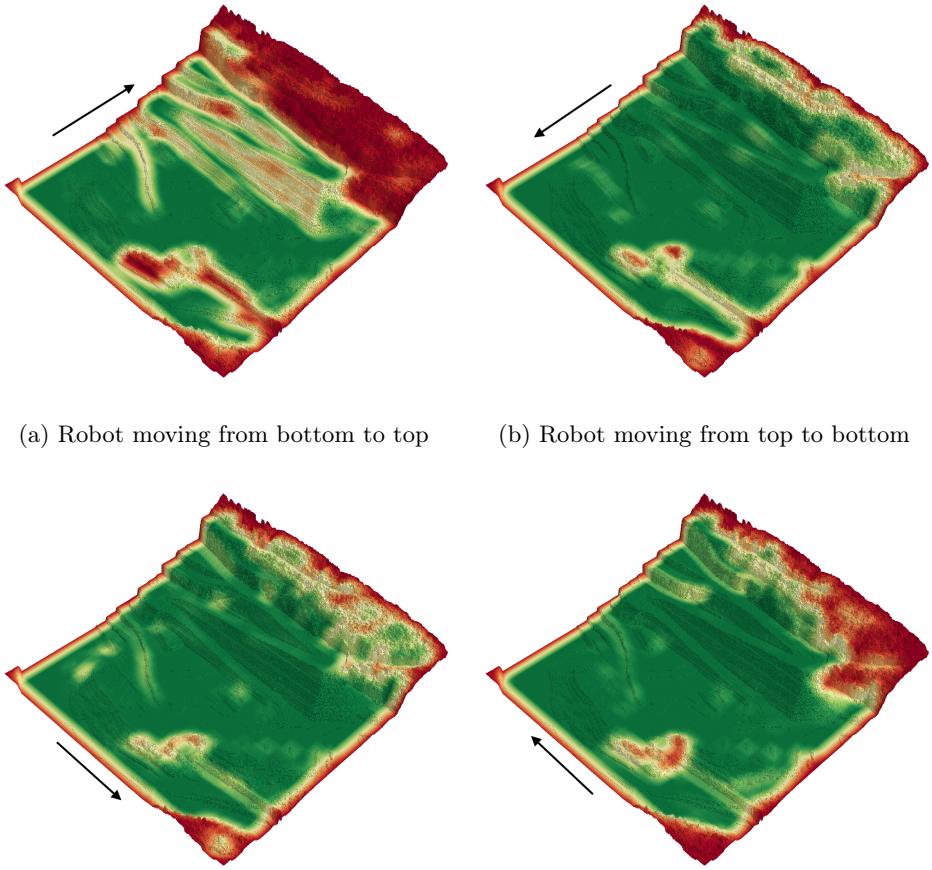
0.3.4 Qualitative results

We qualitative evaluate the model predictions by showing the traversability probability on different maps in 3D as a texture. Specifically, we used a sliding window to extract the patches from the heightmaps and create a texture based on the model's output for the traversable class. Then, we apply this texture on the image and colour using a colormap. For each map we show the traversability from bottom to top, top to bottom, left to right and right to left since those are the most human understandable.

Quarry

The first map we evaluate is Quarry, this map is 32×32m long and has a maximum height of 10m. We can use some of the terrain interesting features, such as are three bis slopes and the rocky ground on top, to evaluate the model. For instance, w expect the trail on the slopes to be traversable at almost any rotation, expecially from left to right and viceversa. While, the top part should be hard to traverse in almost any case. The following figure shows the traversability probability directly on the map.

some where place the colormap bar



(a) Robot moving from bottom to top

(b) Robot moving from top to bottom

(c) Robot moving from left to right

(d) Robot moving from right to left

Figure 1. Traversability probability on the Quarry map, 32×32m, for different Krock's rotation. The values are obtained by sliding a window on the map to create the patches and then predict the traversability for each one of them.

Correctly, the lower part of the map, composed by flat regions, was label with high confidence as traversable in all rotations. On the other hand, the traversability of the slopes and the bumps on the top region depends on the robot orientation.

One oblious example are the slopes on then zig-zag trail. When the robot is moving up-hill, figure 1a, the propability to traverse those steep patches is low. But, if we travel down hill, figure 1b, then the slopes are traversable. This is also true, when the robot is moving from left to right and right to left, figure 1c and 1d is able to travel the slopes since the inclination is not too big to make the robot fall.

Bars

Bars is a map composed by different heights wall, thus we expect similar probabilities with different orientation. For completeness, more images follow with their traversability

probability.

add images of bars, arcs rocks

Sullen

The following map is a small village downloaded from sensefly online datasets hub. Since the map is huge, we crop a

TODO

0.4 Interpretability

In this section, we will evaluate the model's prediction to better understand it. We will find if there are any features in the patches that can confuse it and if the model's output is robust. First, we will introduce on technique used to highlight the region of the input image that contribute the most to the model predictions. Then, we will use it on the data from the *Quarry* test set to find out the patches were the model fails and analyze them.

Later, we will work with custom created patches with different features, walls, bumps, etc, to test the robustness of the model by comparing its predictions to the real data gathered from the simulator.

Grad-CAM

Informally, Grad-CAM [8] is a technique to produce "visual explanations" for convolutional neural networks. It uses the gradient flowing into the last convolution layer to highlight a regions of interested in the image that contribute the most to the final prediction.

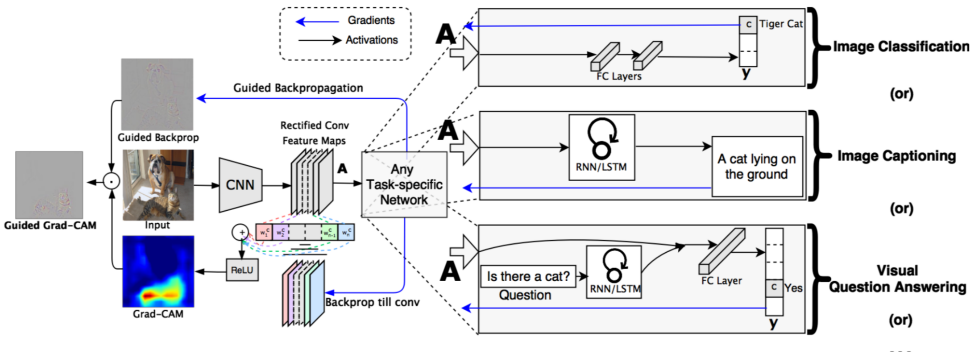


Figure 2. Grad-CAM procedure on an input image [8].

In detail, the output with respect to a target class is backpropagate while storing the gradient and the output in the last convolution. Then global average is applyed the saved gradient keeping the channel dimension in order to get a 1-d tensor, this will represent the importance of each channel in the target convolutional layer. We each element of the convolutional layer outputs is multiplyed with the averaged gradients to create the grad cam. This whole procedure is fast and it is architecture independent.

z