Learning Ground Traversability from Simulations

R. Omar Chavez-Garcia, Jérôme Guzzi, Luca M. Gambardella and Alessandro Giusti

***Abstract*— Mobile robots operating on unstructured terrain must predict which areas of the environment they are able to pass in order to plan feasible paths and to react to unforeseen terrain patterns. We address traversability estimation as a heightmap classification problem: we build a convolutional neural network that, given an image representing the heightmap of a terrain patch, predicts whether the robot will be able to traverse such patch from left to right. The classifier is trained for a specific wheeled robot model (but may implement any other locomotion type such as tracked, legged, snake-like), using simulation data on a variety of procedurally generated training terrains; once trained, the classifier can quickly be applied to patches extracted from unseen large heightmaps, in multiple orientations, thus building oriented traversability maps. We quantitatively validate the approach on real-elevation datasets and implement a path planning approach that employs our traversability estimation.**

Videos and Supplementary Material

Data and code to reproduce our results, video demonstrations, media material, and additional information are available online: `https://github.com/romarcg/traversability_estimation_pioneer`

I. Introduction

In most indoor scenarios, mobile robots are equipped with an internal map of the environment, which is divided into traversable and non-traversable cells. A cell containing movable obstacles or walls is labelled as *non-traversable*, while a cell with no obstacles is labelled as *traversable*. Using this internal map, path planning can be solved using well-known algorithms [1].

In outdoor scenarios, creating a similar map of the terrain might be challenging. A cell might be traversable only in a specific direction, and by a specific robot: a wheeled robot with limited power might be able to descend, but not ascend, a slope; a tall robot with a high center of mass might both ascend and descend the same slope, but then capsize when traversing it from side to side; and a bicycle might hop on a sidewalk, but only if it approaches it from an orthogonal angle. Moreover, it is difficult to anticipate all the difficulties that a robot may encounter: a legged robot may stumble in a terrain with holes of a size comparable to its feet, vacuum cleaner robots might get stuck over power cords, a car with a low chassis may not pass over a speed bump.

We consider the problem of estimating where and in which directions a given 3D terrain is traversable by a specific ground robot, using a general approach based on machine learning that applies regardless of the robot's locomotion method (wheeled, tracked, legged, snake-like), physical characteristics (size, motor torque), and low-level controller (anti-skid algorithms for wheeled robots, foothold selection and gait selection algorithms for legged robots). We will define a given terrain patch as *locally traversable* in a given direction if the robot, once placed in the center of the patch, can proceed in that direction for at least a short distance when driven by its low-level control algorithm.

When robot control parameters are known, traversability is only affected by the characteristics of the terrain around the robot's position. Therefore, for a given robot pose X^{robot} with position p and orientation θ, we consider as input a heightmap patch centered in p and rotated in such a way that the robot is pointing towards the right of the patch. The patch is represented as a heightmap image where pixels indicate height values. Our task is cast as a binary image classification problem (with classes *traversable* vs *non-traversable*), and solved by training a simple convolutional neural network.

Training datasets are generated by simulating the robot on many procedurally generated training terrains, which represent a variety of obstacles such as ramps, steps, bumps, holes and rugged areas; during such simulations, the robot is spawned in random positions and orientations, and instructed to proceed straight ahead; its progress is monitored and instances for traversable (where the robot successfully proceeds) and non-traversable (where the robot can't proceed) heightmap patches are continuously recorded. Once the model is learned from such training data, it can be applied densely on any large unseen heightmap. Because accurate physical simulation is expensive, evaluating the classifier on many points and orientations of a test terrain is orders of magnitude faster than simulating the robot on these poses.

After a brief review of the literature (Section II), we introduce the **main contribution** of this work: a complete framework for traversability estimation (detailed in Section III). It includes i) an algorithm for procedural generation of training heightmaps, ii) an heightmap-patch classifier, iii) an orientation-based traversability representation, and iv) an application of such representation to path planning in real heightmaps and with a real robot. Experimental validation and results are described in Section IV.

II. Related Work

Estimating traversability is a fundamental capability for many animals and for autonomous mobile robots, because most of their actions depend on their mobility. Traversability can be seen as an *affordance* of the environment on the

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Research (NCCR) Robotics.

R. Omar Chavez-Garcia, Jérôme Guzzi, Luca M. Gambardella and Alessandro Giusti are with the Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, Lugano, Switzerland. `omar@idsia.ch`

robot, or vice-versa [2], the former defines which terrain characteristics allow the robot to traverse, the latter includes robot's capabilities (e.g. morphology, motor, sensing) to traverse the environment. In robotics, using a simplified (possibly learned) model to estimate traversability is a common approach, because modeling the terrain, the robot and their interaction accurately is difficult and expensive. Several approaches have been proposed to measure traversability and to gather training data [3]. For instance, a robot may label a terrain as difficult to traverse when sensing excessive vibrations [4], [5]; human experts can provide clues like preferred paths in a given terrain that avoid possible non-traversable regions [6]; the robot may learn a traversability classifier on the spot, directly from experience [7].

The outdoor robotics community has researched traversability estimation from a variety of sensing sources, such as stereo cameras [8], laser scanners [9], and time of flight cameras [10]. A common approach to estimate traversability consists of two phases. First, from local sensory data, an elevation map is derived , which is a convenient spatial representation for ground robots [11]. Then, traversability estimates of parts of the elevation map are computed from geometrical features like *slope*, *roughness*, and *step height* using pre-modeled functions specific to the robot's locomotion and size [12].

Another major approach relies on the use of visual texture to classify the terrain type, e.g. discriminating between rock, sand, and grass; then it derives a traversability score out of the assigned label [4]. Terrain classification is a classical application for supervised machine learning techniques based on generic visual features, transformation-invariant descriptors, texture features, or convolutional neural networks (CNNs) [13], [14], which learn to extract problem-specific visual features.

DARPA's project *Learning Applied to Ground Robots project* [15] has advanced learning techniques for terrain traversability classification, including some applications of deep learning [16]. In this regard, CNNs have been used to rapidly learn on-spot terrain classification from areal images collected by autonomous drones [17] and deep CNN have enabled autonomous drones to follow forest trails based on a frontal camera's video stream [18].

III. Traversability Estimation Framework

Figure 1 illustrates the proposed framework for traversability estimation. Next, we describe how the simulation environment is set-up (Section III-A), detail the process for generating training and evaluation datasets (Section III-B), and finally describe classification approaches (Section III-C).

A. Traversability from Simulation

We employ the Gazebo simulator and the ODE physics engine [19] for accurate physical simulation. We simulate the Pioneer 3-AT robot platform of size $49 \times 50 \times 29$ cm (illustrated in Fig. 1) moving forward at a constant velocity of $0.15m/s$ on an uneven terrain whose shape is determined by a heightmap. The robot's trajectory on the terrain is captured to extract traversability information.

1) Heightmap Generation: In order to generate meaningful training data, we need to simulate the robot moving on interesting terrains that pose varied and representative challenges. This could be achieved by using real heightmap data, by creating interesting terrains manually, and/or by synthesizing them using procedural generation techniques. We follow exclusively the third option. Each of our training terrains is generated as a grayscale heightmap, based on summing multiple realizations of random 2D simplex noise [20], [21]: a variant of Perlin noise [22], conceptually similar to it, frequently adopted in the procedural generation literature [23]) with different periods.

For example, a heightmap, corresponding to simplex noise with period 30 cm and scaled such that it extends to a height of 20 cm, models a terrain with medium-sized smooth rocks while a period of 10 m with a 3 m height range yields a landscape with small steep hills. By summing the two heightmaps, one obtains rocky hills. If the *rocks* heightmap is multiplied by a value between 0 and 1 which increases on the x coordinate, one gets flat hills for low values of x which becomes rocky hills for high values of x. Additional features such as holes, bumps protruding from a flat surface, rail-like indentations, steps, and others, can be generated by applying various functions on each value in the heightmap. Some example training terrains are represented in Fig. 2. The size of each generated heightmap is 512×512 px that, when simulated, is scaled to represent a surface of 10×10 m (≈ 2 cm/px resolution).In the supplementary material we provide source code for generating our training dataset and an appendix describing the approach in more detail.

2) Simulation Process: Once the simulation is initialized with a generated heightmap, the robot is set to a random pose on the map and moves forward at constant velocity without steering. After the robot reaches the edge of the map or gets stuck for some time, it is re-spawned to a different pose to generate a new trajectory.

For each trajectory, the robot poses and their corresponding heightmap patches are recorded; the heightmap patch associated to a pose is centred on the robot's position and oriented in such a way that the robot is facing towards the right of the patch (see Fig. 1-middle top). If and only if the distance between the current pose $X^{\text{robot}}(t)$ and a future pose $X^{\text{robot}}(t+T)$ is greater than a threshold d and aligned with the robot's orientation, then the current patch is labelled as traversable. Otherwise the current patch is labelled as non-traversable.

Figure 3-left illustrates the traversability labelling for patches along a trajectory, with $T = 1$s and $d = 0.12$m. Following this trajectory the robot is able to traverse a set of downward-sloped patches until it gets stopped by a bump almost as high as the robot itself. This behaviour is represented in Fig. 3-right, where traversable patches are marked with green frames.

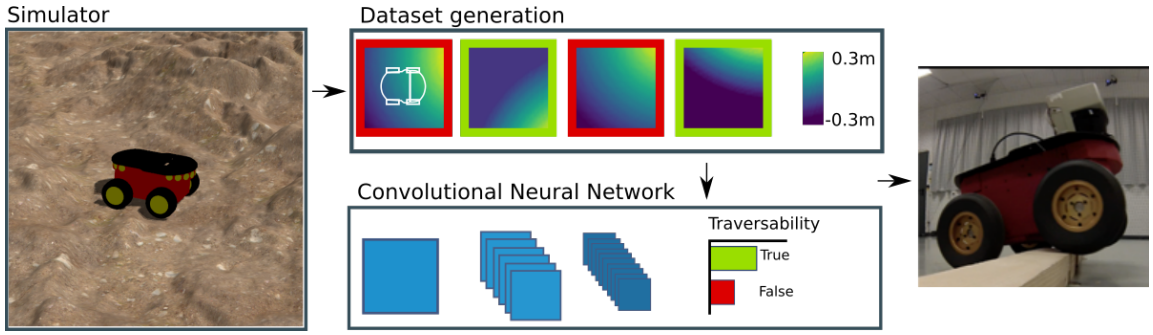

Fig. 1. Robot model runs in simulation on procedurally generated terrains (left) to generate datasets linking heightmap patches with their traversability (top); on these datasets we train classifiers to estimate the probability that a given heightmap patch is traversable or not (bottom). The learned classifier correctly predicts terrain traversability for a real robot (right).

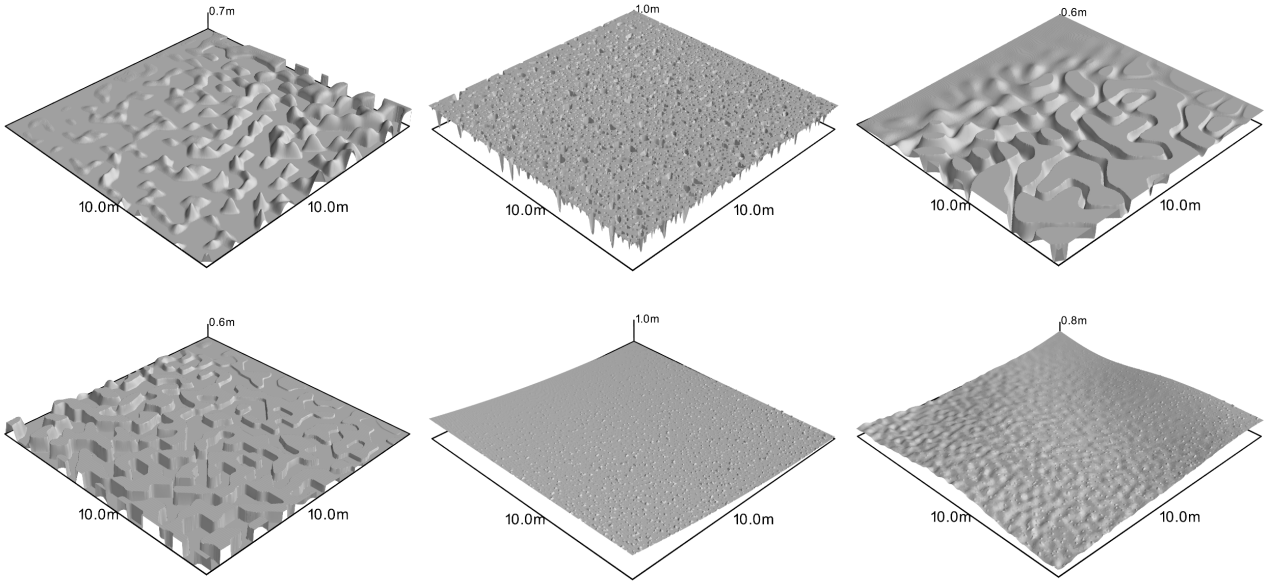

Fig. 2. Examples of heightmaps for generating traversability datasets. These maps includes common terrain varieties such as (from top-left to bottom-right): bumps, holes, rails, steps, slopes, and a combination of them.

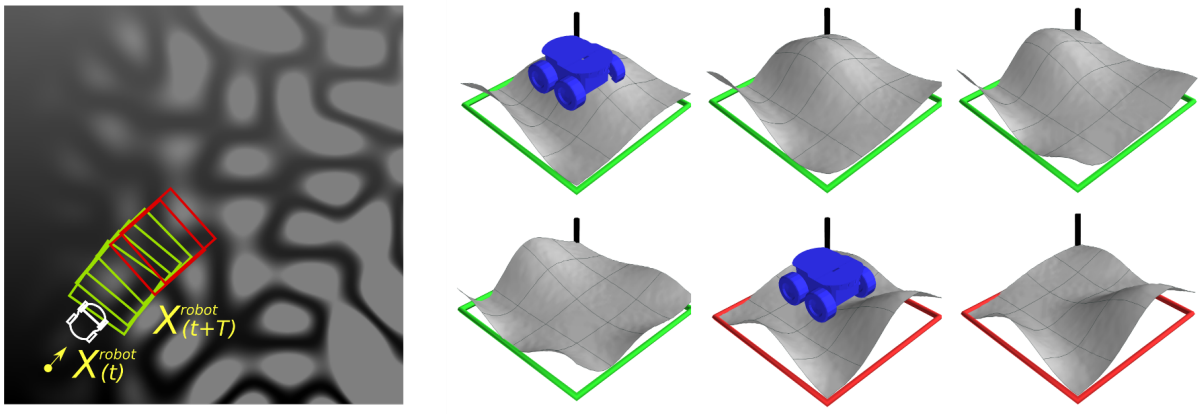

Fig. 3. Trajectory extracted from a synthetic dataset (left). Robot silhouette and yellow arrow indicates the initial pose for the corresponding trajectory. Patches (represented as surfaces) labelling from the same trajectory (right). Green frames indicate patches labelled as traversable, red frames as non-traversable.

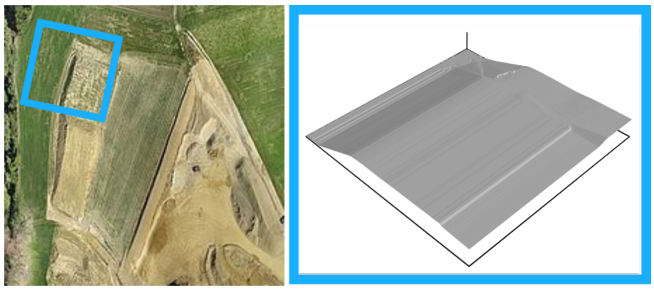

Fig. 4. Sample surface from a gravelpit elevation data. For generating the gravelpit dataset, several heightmaps were extracted from a mapping of a Swiss gravel pit [24]. Intensity represents height.

B. Dataset Generation

From each simulated trajectory (on average 20 s), we sample the robot pose and the corresponding heightmap patch at 20 Hz. The patch and its traversability label represents an instance in the dataset.

We generate four datasets: D_{train}, a training dataset (450k samples) from procedurally generated heightmaps (see Section III-A.1); $D_{\text{eval,syn}}$, an evaluation dataset (150k samples) with synthetic heightmaps also procedurally generated; $D_{\text{eval,gravelpit}}$, an evaluation dataset from a real gravel pit map (60k samples); and $D_{\text{eval,quarry}}$, an evaluation dataset from a real mining quarry map (100k samples). Considering the average trajectory length of 0.06m travelled by the simulated robot in each independent sample, the training dataset is equivalent to traversing 27km. Likewise, the evaluation dataset is equivalent to traversing 18.6km.

The maximum height of the heightmaps is 1m for D_{train} and $D_{\text{eval,syn}}$; 3m for $D_{\text{eval,gravelpit}}$; and 10m for $D_{\text{eval,quarry}}$. All datasets were generated using the robot described in Sec. III-A, a patch size of 60×60px, $T \approx 1$s and $d = 0.12$m.

Figure 3-right illustrates patches from the trajectory of Fig. 3-left. The height values in each patch are offset in such a way that the patch center (i.e. at robot's position) is mapped to height 0. This makes the patches independent on their absolute height on the heightmap, a feature that does not affect traversability.

Dataset $D_{\text{eval,gravelpit}}$ is generated using an elevation map from a Swiss gravel pit obtained by a flying drone [24]. The original area of this map is 0.48km^2, with a resolution of 5cm/px and a maximum height of 50m. We cropped and scaled two regions of this map to form two heightmaps of 10×10m with a resolution of 1.9cm/px and a maximum height difference of 3m. Figure 4 shows a reference image of the gravel pit and the surfaces of two extracted heightmaps.

C. Training Traversability Classifiers

We address the problem of estimating terrain traversability as classification on heightmap data. We compare two alternative approaches: extracting descriptive features from each heightmap patch and then applying standard statistical classification techniques [25], or adopting Convolutional Neural Networks, a now-standard deep-learning approach [26] which operates directly on the raw input data. In either case, the output of the classifier indicates whether a patch is traversable.

For the **feature-based approach** we should extract from the input heightmap patch some quantities that provide traversability cues, for example the average terrain steepness in the robot's motion direction (i.e. from left to right of the patch), or the maximum height of any steps in patch. In our case, we compute the Histogram of Gradients (HOG) of the heightmap patch, which includes these pieces of information (e.g., the gradient of a heightmap corresponds to the local steepness). Computing HOG over 6 orientations, 8×8px per cell, and a block of 3×3 cells, results on a descriptor with 324 features that we classify by means of a Random Forest (RF) classifier [27] with 10 trees.

In the **CNN-based approach**, it is expected that the network autonomously learns meaningful, problem-specific features; because the input data is high-dimensional and no prior knowledge of the problem is provided to the model, this approach requires more training data, which however is available from our extensive simulations. Our CNN is built on the Keras [28] frontend powered by TensorFlow [29]; it is composed by interleaved convolutional and max-pooling layers, a classic architecture that is well known to be suited to visual pattern recognition problems [30]; a 60×60 px input layer is followed by: a 3×3 convolution layer with 5 output maps; a 3×3 convolution layer with 5 output maps; a 2×2 Max-Pooling layer; a 3×3 convolution layer with 5 output maps; a fully connected layer with 128 output neurons; a fully connected layer with 2 output neurons followed by a softmax layer (output). All layers implement the ReLU activation function. The network is trained for 100 epochs to minimize a categorical cross-entropy loss using the Adadelta optimizer. Following the current best practices for visual pattern recognition CNNs [28], several variations on this architecture have been evaluated with minimal performance differences.

IV. Experimental Results

The following sections detail the comparison of the two classifiers described in Section III-C and an additional baseline dummy classifier that always returns the class most frequent in the training set.

A. Classification Results

Performance of the three estimators on the two different evaluation datasets (disjoint from the training set) $D_{\text{eval,syn}}$ and $D_{\text{eval,gravelpit}}$ is presented in Table I. We observe that the CNN estimator outperforms both the baseline and feature-based approaches on both evaluation datasets. Performance is lower on the real datasets $D_{\text{eval,gravelpit}}$ and $D_{\text{eval,quarry}}$ than on $D_{\text{eval,syn}}$, probably because of some elevation patterns, such as narrow sections, tight rails and low barriers that may block the robot, which are not represented in procedurally-generated training data. Nonetheless, many features of real terrains such as slopes, holes, rails, steps and bumps are correctly classified.

	$D_{\text{eval,syn}}$		$D_{\text{eval,gravelpit}}$		$D_{\text{eval,quarry}}$	
	ACC	AUC	ACC	AUC	ACC	AUC
CNN	0.926	0.970	0.832	0.839	0.819	0.840
Feature-based	0.703	0.746	0.769	0.802	0.723	0.762
Baseline	0.544	0.497	0.577	0.504	0.542	0.499

TABLE I

PERFORMANCE OF CNN, FEATURE-BASED AND BASELINE APPROACHES. ACCURACY (ACC) AND AREA UNDER THE ROC CURVE (AUC) METRICS ARE FROM A SYNTHETIC DATASET $D_{\text{EVAL,SYN}}$ AND TWO REAL DATASETS: $D_{\text{EVAL,GRAVELPIT}}$ AND $D_{\text{EVAL,QUARRY}}$.

B. Traversability Visualization on the Quarry Dataset

The quarry dataset was generated from a mining quarry map (see Fig. 5) of 0.51km^2 [24]. This map contains challenging roads and barriers designed for mining trucks. To make the map more suitable for our robot, which is about 0.05x the size of a mining truck, we rescaled the map to 0.05x its original size.

For this analysis, we fix a direction and iterate over the entire heightmap extracting patches of 60×60px with a stride of 5px. This process is equivalent to translating the robot's position over the map while keeping a fixed orientation. Figure 6 shows the traversability estimation for the mining quarry for four orientations, indicated by the arrows. Traversability is represented as a colored overlay on the surface of the heightmap (traversable is green, non-traversable is gray). The minimum traversability overlay (blue frame) is calculated as the the minimum traversability estimation at each position from a set of possible orientations (in this case 32). In some positions (gray areas) such minimum value is 0. This representation shows a general view of the traversal capabilities of a robot in a particular heightmap.

The estimator correctly marks the main road (≈ 1.9m wide) as traversable in all directions. Very steep or vertical slopes are recognized as non-traversable when going up but, dangerously, marked traversable when going down. This particular result is congruent with the definition of traversability we used, which only considers whether the robot can proceed but does not account for its safety. Slopes are always classified as traversable downhill and, sometimes, transversally. Most of the rough surface at the top of the elevation map is correctly found as non-traversable. The minimum traversability representation (Fig. 6-right) indicates only areas that the robot can traverse in any of 32 orientations. These, correctly, correspond to roads and flat terrain.

C. Traversability Estimation Performance

Uniform terrains, such as plains and low-hills, do not represent a challenge when estimating traversability, a simulation could directly provide their traversability in real-time. However, complicated terrains in real scenarios (search and rescue areas, countryside exploration, clutter-environment navigation) increase the computing requirements to properly calculate simulation parameters. Considering the quarry map presented in Fig. 6, most of its terrain contains a combination of several patterns: slopes with steps, rails and holes. When simulating the patches of this terrain (without model rendering), the average real-time factor was ≈ 0.1. The time needed by our trained network to estimate the traversability of a patch was 0.16ms.

D. Path Planning with Probabilistic Traversability Estimation

Traversability maps can be used to plan a path to a goal position which only passes through patches which are traversable in the direction corresponding to the path's local orientation. In the following, we assume that the robot will get stuck (or crash, or end up in an unrecoverable state) if it attempts to pass through a non-traversable patch[1].

We assume that traversability probabilities of non-overlapping patches are independent, i.e. that the traversability along a path is Markovian. This allows us to compute the probability p to traverse a path composed of segments $(e_1, \ldots, e_n)$ as $p((e_1, \ldots, e_n)) = \prod_{i=1}^{n} p(e_i)$, where $p(e_i)$ is the probability to traverse a segment.

We define a directed graph (N, E) where: nodes in N are placed on a regular square grid of side length 18 cm, and; edges in E connect each node with its 8 nearest neighbors. For each edge $e \in E$ we compute length and traversal probability $p(e)$. $p(e)$ is the output of the traversability classifier applied to a patch centered at the edge's origin and oriented along the edge direction (see Fig. 7-right).

Figure 7-right bottom illustrates the solution of the multi-objective problem of computing the best paths with respect to length (minimized) and traversability (maximized). A rational agent would choose among the set of Pareto-optimal paths. In this context, a path is Pareto-optimal if there exists no alternative path that is both more traversable and shorter. The shortest path (represented in red in Fig. 7-left) is Pareto-optimal, but will have in most cases a very low traversability probability. The path with the maximum traversability (represented in green) will also be Pareto-optimal, but may be unnecessarily long; between the two extremes, a potentially very large set of Pareto-optimal paths exist, spanning the trade-off between traversability and length.

a) Real robot experiment: We test how our approach applies to a real Pioneer 3-AT robot in an environment where a pair of bars (1m length $\times$ 6cm height) of width 6cm and 8cm respectively lie horizontally aligned on a flat floor with a small gap of 15cm in-between (see Fig. 8); the bars are fixed to the floor. We use a hand-generated heightmap representing the known geometry of the scene. The robot is tasked to move from the area below the leftmost bar to the area above it. There is no space to pass left of the bars, but plenty of space to pass on the right of the map. The robot is constrained to move along vertical paths over (or beside) the obstacles. To verify whether a given path is traversable, we teleoperate the

[1]This is a pessimistic assumption because this is not always the case: a robot attempting pass through a wall, for example, is unable to proceed but may be capable of turning back and taking another path.

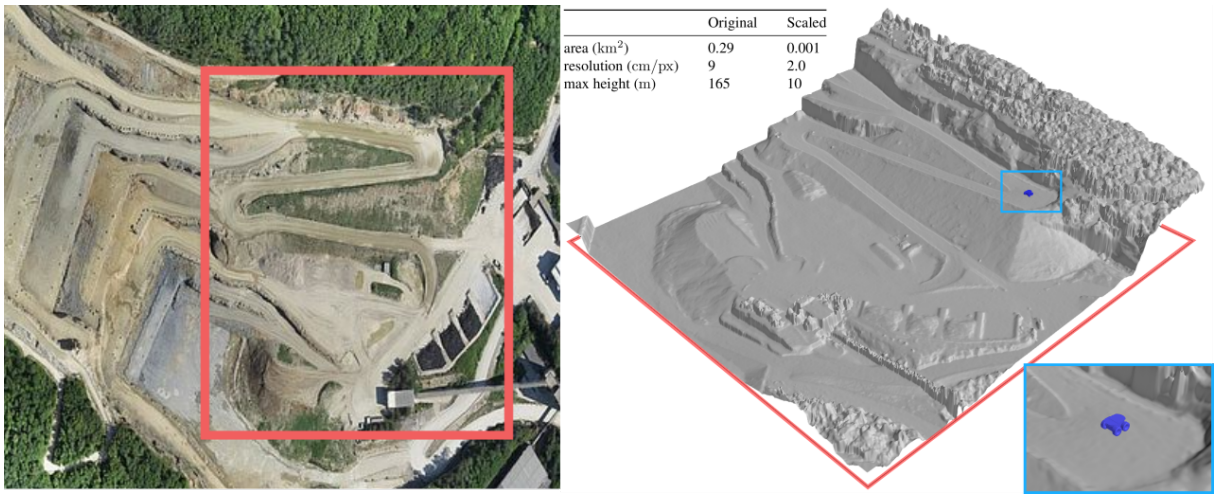

Fig. 5. Heightmap extracted from Sensefly's mining quarry dataset [24]. Reference top-view image and perspective view of the elevation map from the extracted red region are shown at left and right respectively. Pioneer 3-AT's model is displayed for size comparison.

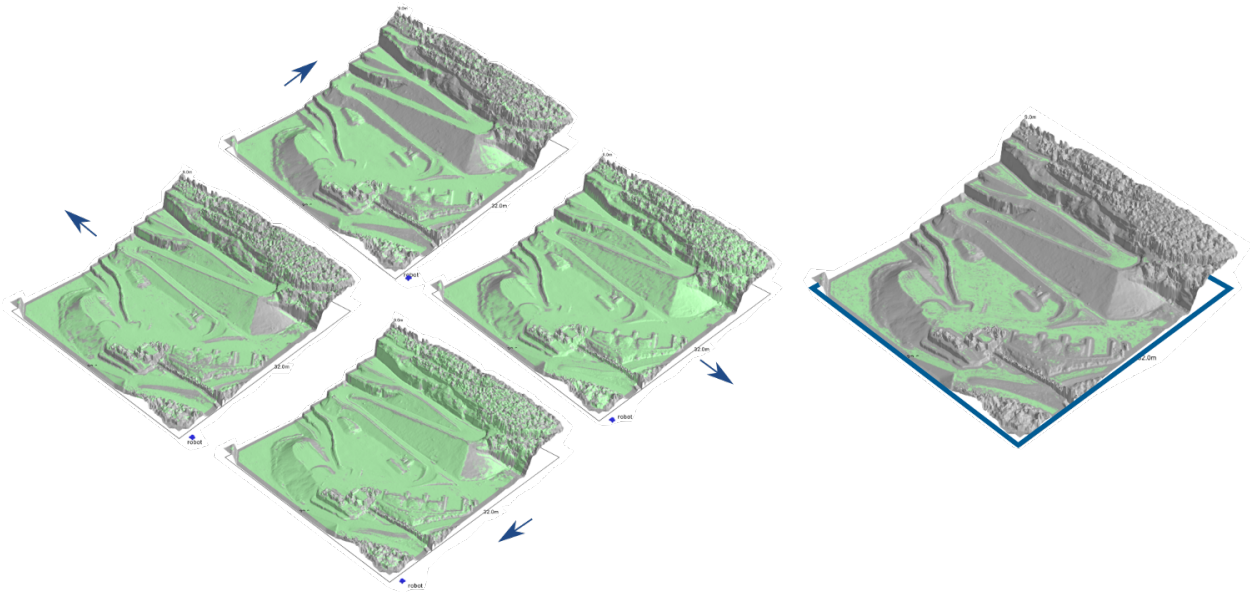

Fig. 6. Left: Traversability estimation for the mining quarry elevation map, for four orientations: 180° (left), 90° (up), 270° (down), 0° (right). Right: Minimum traversability (blue frame) overlay over 32 possible orientations. For all figures, green indicates traversability and gray indicates non-traversability.

robot following such path. We test three different trajectories: the most direct one where the robot will always get stuck (red); the one when it may pass in the space between the two bars by having wheels on one side climbing the bars, whereas the wheels on the other side passing in the gap, this path has a probability of about 50% that was determined empirically through several repetitions; and the longer but safest one going around the obstacles (green).

V. Conclusions and Perspectives

We presented a complete framework for traversability estimation that casts the problem as an elevation-map classification task. Classifiers trained on simulation data capture complex characteristics of different robot models and quickly estimate traversability maps on large unseen terrains.

We assumed that the terrain's 3D shape is the only factor influencing traversability; other factors may also be considered, such as compactness, friction, and instability. Our framework could handle these factors provided they can be simulated: for example, if terrains whose 3D shape suggests a loose gravel surface were simulated with lower friction, the classifier would automatically learn that steep slopes can't be negotiated on such surfaces; in this perspective, one may add additional inputs to the classifier (such as the visual texture) which may help differentiate the actual surface.

Our current approach does not capture the robot dynamics, such as the speed with which the robot approaches an obstacle: we limit our attention to slow robots operating on rugged terrains, where dynamic aspects have negligible

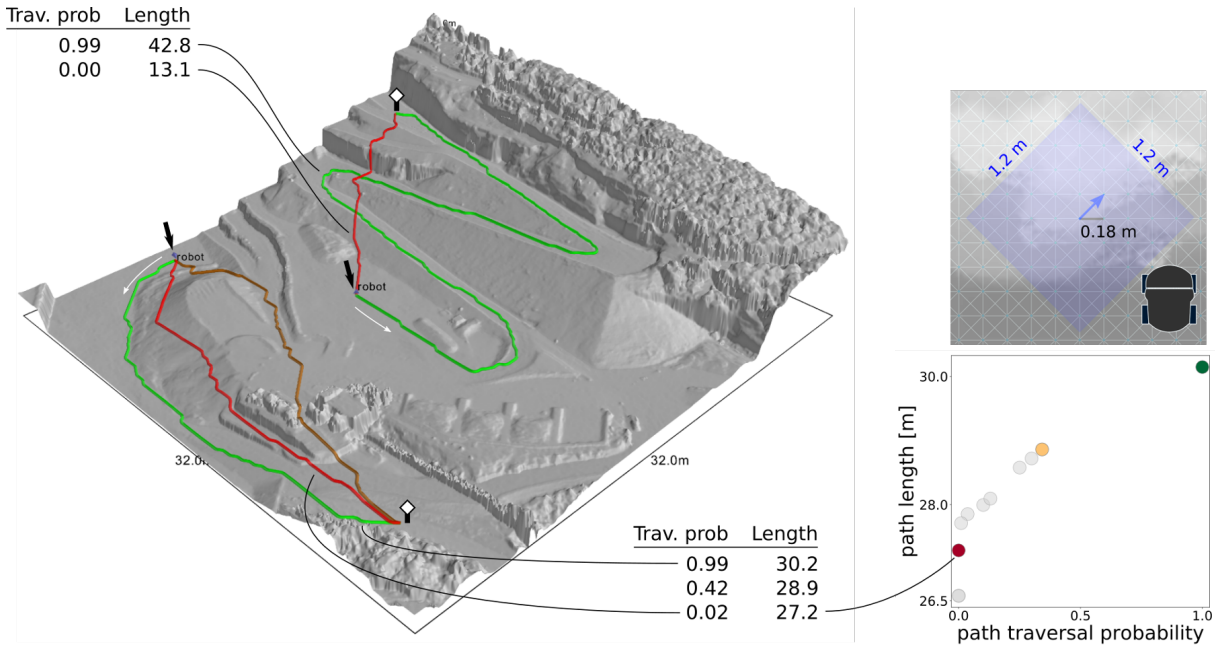

Fig. 7. Left: a selection of Pareto optimal paths on the quarry map for two pairs of source (arrow) and target (white square) locations. The paths are coloured by traversal probability from red (non-traversable) to green (surely traversable). Corresponding values for traversal probability and length of each trajectory are shown in the side tables. Right top: a portion of the planning graph on the quarry map. Nodes are placed on a regular grid at 18 cm. The blue edge's traversal probability is estimated by applying the classifier on a 1.2m x 1.2m patch (light blue) centered at the edge origin and directed along the edge. Robot's silhouette is shown for size comparison. Right bottom: The trade-off between path length and traversability for Pareto optimal paths of the bottom source-target location. Colored dots correspond to paths drawn on the left figure.

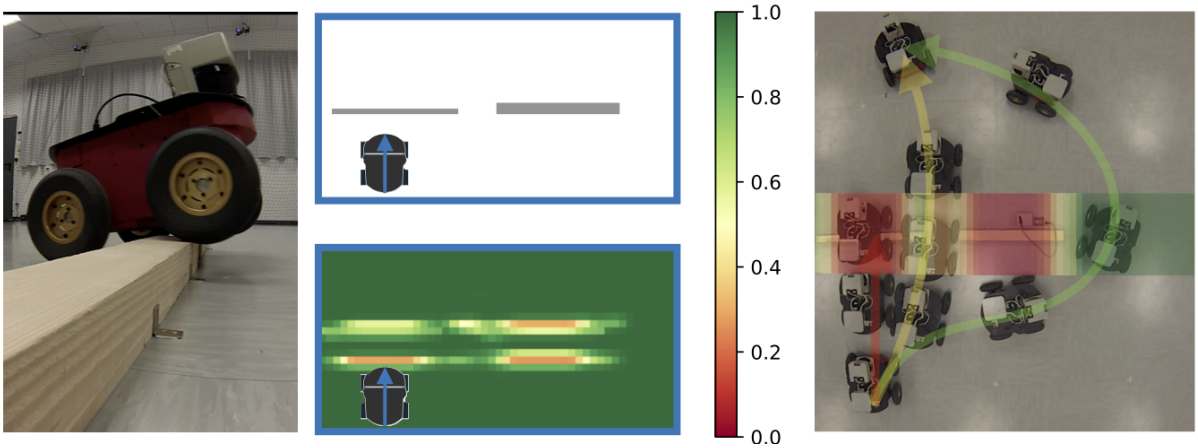

Fig. 8. Left: Pioneer 3AT tries to pass over the bars. Center: height-map with two bars of width 6 cm and 8 cm (top) and patch traversability estimation moving upwards (bottom). Right: robot tries alternative routes. For all paths going straight upwards, the estimated traversal probability is displayed on top of the bars.

impact. We consider the addition of such information as we move towards scenarios where complex locomotion is needed.

References

[1] R. Hadsell, M. Scoffier, U. Muller, Y. LeCun, and P. Sermanet, "Mapping and planning under uncertainty in mobile robots with long-range perception," in *In Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008, pp. 1–6.

[2] U. E. and S. E., "Traversability: A case study for learning and perceiving affordances in robots," *Adaptive Behavior*, vol. 18, pp. 258–284, 2010.

[3] V. Molino, R. Madhavan, E. Messina, A. Downs, S. Balakirsky, and A. Jacoff, "Traversability metrics for rough terrain applied to repeatable test methods," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2007, pp. 1787–1794.

[4] C. A. Brooks and K. D. Iagnemma, "Self-supervised classification for planetary rover terrain sensing," in *2007 IEEE Aerospace Conference*, March 2007, pp. 1–9.

[5] M. A. Bekhti, Y. Kobayashi, and K. Matsumura, "Terrain traversability analysis using multi-sensor data correlation by a mobile robot," in *2014 IEEE/SICE International Symposium on System Integration*, Dec 2014, pp. 615–620.

[6] D. Silver, J. A. Bagnell, and A. Stentz, "Learning from demonstration for autonomous navigation in complex unstructured terrain," *Int. J. Rob. Res.*, vol. 29, no. 12, pp. 1565–1592, Oct. 2010. [Online].

Available: http://dx.doi.org/10.1177/0278364910369715

[7] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, J. Han, U. Muller, and Y. LeCun, "Online learning for offroad robots: Spatial label propagation to learn long-range traversability," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[8] A. Chilian and H. Hirschmuller, *Stereo camera based navigation of mobile robots on rough terrain*, 2009, pp. 4571–4576.

[9] M. Herbert, C. Caillas, E. Krotkov, I. S. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *Proceedings, 1989 International Conference on Robotics and Automation*, May 1989, pp. 997–1002 vol.2.

[10] H. Balta, G. D. Cubber, D. Doroftei, Y. Baudoin, and H. Sahli, "Terrain traversability analysis for off-road robots using time-of-flight 3d sensing," in *In Proc. of the Int. Workshop on Robotics for Risky Environment - Extreme Robotics*, 2013.

[11] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila, "Autonomous rover navigation on unknown terrains functions and integration," in *Experimental Robotics VII*, ser. ISER '00. London, UK, UK: Springer-Verlag, 2001, pp. 501–510. [Online]. Available: http://dl.acm.org/citation.cfm?id=645627.757994

[12] M. Schwarz and S. Behnke, "Local navigation in rough terrain using omnidirectional height," in *ISR/Robotik 2014; 41st International Symposium on Robotics*, June 2014, pp. 1–6.

[13] R. Hudjakov and M. Tamre, "Aerial imagery terrain classification for long-range autonomous navigation," in *2009 International Symposium on Optomechatronic Technologies*, Sept 2009, pp. 88–91.

[14] M. Längkvist, A. Kiselev, M. Alirezaie, and A. Loutfi, "Classification and Segmentation of Satellite Orthoimagery Using Convolutional Neural Networks," *Remote Sensing*, vol. 8, no. 4, 2016.

[15] L. D. Jackel, E. Krotkov, M. Perschbacher, J. Pippine, and C. Sullivan, "The darpa lagr program: Goals, challenges, methodology, and phase i results," *Journal of Field Robotics*, vol. 23, no. 11-12, pp. 945–973, 2006. [Online]. Available: http://dx.doi.org/10.1002/rob.20161

[16] M. Shneier, T. Chang, T. Hong, W. Shackleford, R. Bostelman, and J. S. Albus, "Learning traversability models for autonomous mobile vehicles," *Autonomous Robots*, vol. 24, no. 1, pp. 69–86, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10514-007-9063-6

[17] J. Delmerico, A. Giusti, E. Mueggler, L. M. Gambardella, and D. Scaramuzza, "On-the-spot training for terrain classification in autonomous air-ground collaborative teams," in *International Symposium on Experimental Robotics (ISER)*, 2016.

[18] A. Giusti, J. Guzzi, D. C. Cireşan, F. L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, July 2016.

[19] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sept 2004, pp. 2149–2154 vol.3.

[20] K. Perlin, "Improving noise," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 681–682.

[21] M. Olano, "Modified noise for evaluation on graphics hardware," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. ACM, 2005, pp. 105–110.

[22] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D. S. Ebert, J. P. Lewis, K. Perlin, and M. Zwicker, "A survey of procedural noise functions," in *Computer Graphics Forum*, vol. 29, no. 8. Wiley Online Library, 2010, pp. 2579–2600.

[23] R. M. Smelik, K. J. De Kraker, T. Tutenel, R. Bidarra, and S. A. Groenewegen, "A survey of procedural methods for terrain modelling," in *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*, 2009, pp. 25–34.

[24] SenseFly, "Elevation datasets," https://www.sensefly.com/drones/example-datasets.html, 2016. [Online]. Available: https://www.sensefly.com/drones/example-datasets.html

[25] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[26] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135

[27] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[28] F. Chollet, *Deep Learning with Python*. Manning Publications Company, 2017. [Online]. Available: https://books.google.ch/books?id=Yo3CAQAACAAJ

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[30] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.