# Chapter 1

# Methodology

We develop a framework to learn and predict traversability that learns directly on ground patches for any robot using purely simulated data. The main idea is to run a robot into a simulator on different synthetic terrain and later used its interactions to train an estimator. Simulation data has two main advantages: variety, cost, and speed. With simulations, we can easily increase the number of data samples by just include more maps, while doing the same with a real robot requires to first identify the suitable ground and then travel to it. Clearly, this also decreases the time required to collect the robot's interaction with the ground. The time could be additional decreased by running different simulations in parallel avoiding physical manufacture or purchase additional robots. Moreover, we are not limited by real constraint and therefore the robot can interact with a multitude of different grounds. For instance, to collect real-world data the robot must be physically transported to different locations, on the other hand, with simulations we can immediately load a new map.

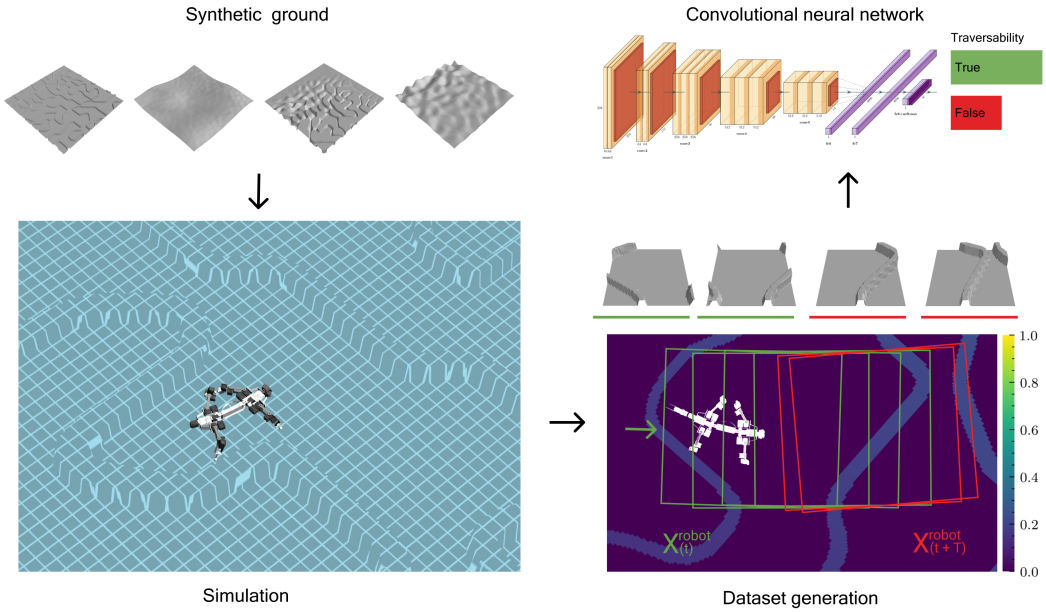
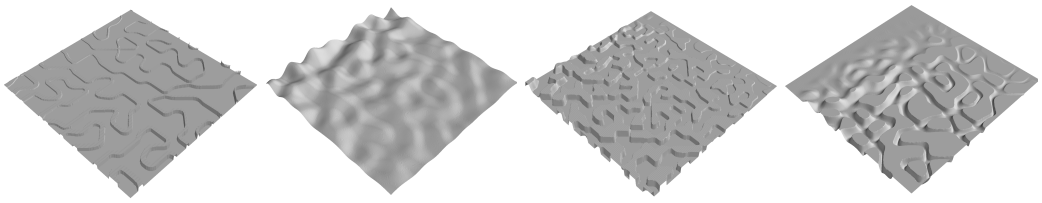



Figure 1.1. The proposed traversability framework's main blocks in couter-clockwise order. First we generated meaningful synthetic ground, then we let the robot spawn and walk on them in simulated enviroment while storing its interactions. Later, we crop a region of ground, a patch, for each simulation trajectory around the robot according to its locomotion. We label those images using a defined treshold and fit a deep convolutional neural network to predict traversability.

TTo create a quality dataset, we must first generate a series of various surfaces and ensure their diversity. Intuitively, those maps should be small enough to allow fast interactions with their features but big enough to include variety. With modern ground generation techniques, it is fairly straightforward to generate a rich array of terrains with different characteristics such us bumps, ramps, slopes in different levels and sizes. These images are stored as gray images where the brightness represent the height.



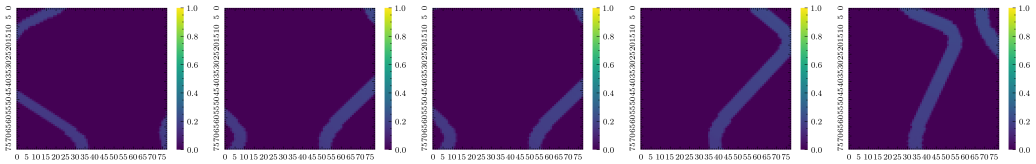
(a) Walls. (b) Bumps. (c) Steps. (d) Rails.

Figure 1.2. Some artificially generated terrains, $10 \times 10$m, with one specific feature each.

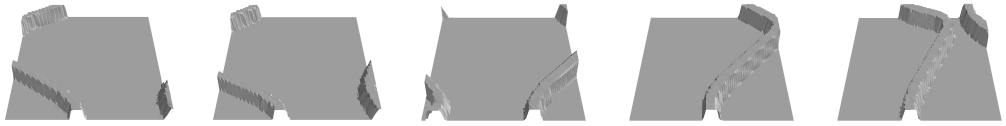
The maps can be loaded into a simulator to let the robot interact with them. To ensure a correct exploration of each terrain, it is better to randomly spawn the robot multiple

times on the same map and let it walk for a fixed amount of time. The robot position is tracked and stored during simulation in order to extract a small portion of ground, patches, around each of its trajectory's position. In practice, the patches are cropped from the original image using a specific size based on the robot's footprint and its locomotion. Each resulting image is label using a minimum advancement defined for the robot. This dataset is feed to a deep convolutional neural network. The network directly learns to extract and map features from raw data, in our case images, without needing to preprocess the inputs.


(a) Robot's trajectory.



(b) Cropped patches in 2D.



(c) Cropped patches in 3D.

Figure 1.3. Example of a robot trajectory (a) extracted during training on a map with different walls. Robot's initial position is showed by its white siluette. Patches borders are label with greed if traversable and red if not and showed as 2D (b) and 3D(c) rendered images.