0.1 Quarry dataset

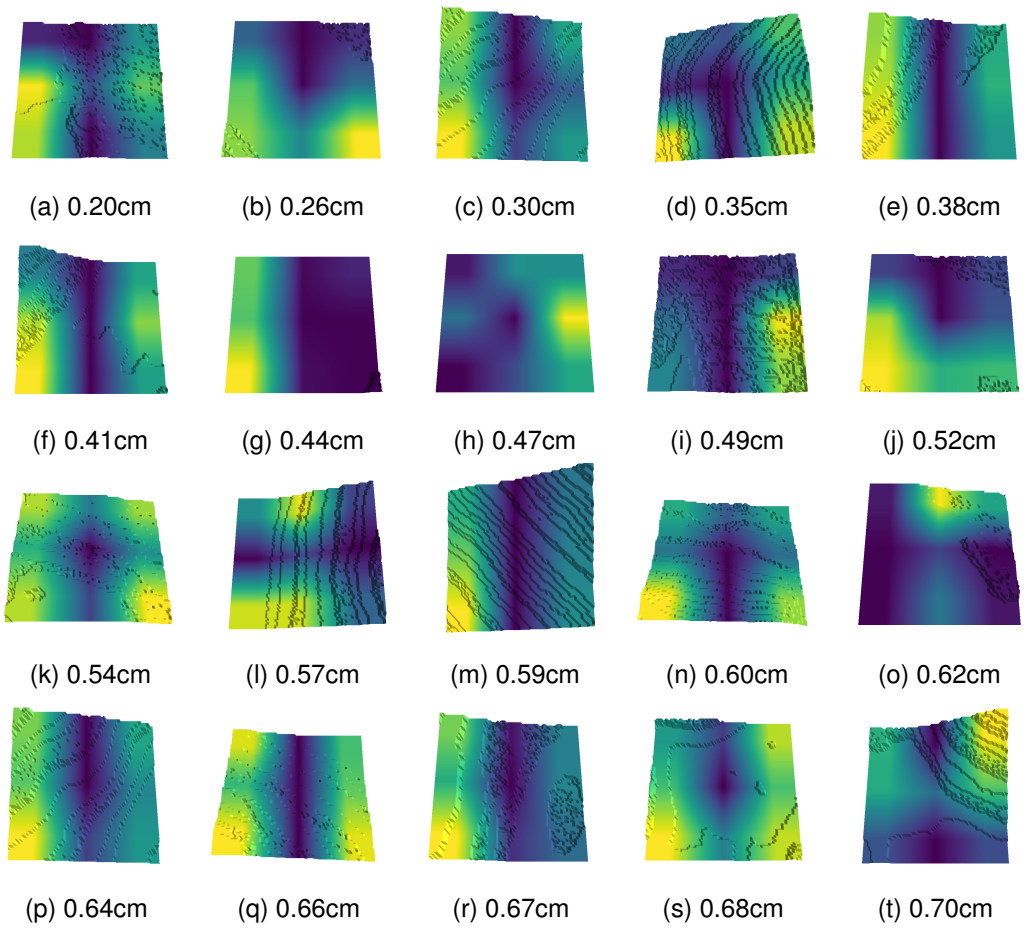
After showing the model's capability of correctly separate classes' features we utilized Grad-CAM to visualize some of the samples in the test set. The aim of this section is to show how the model looks at meaningfull features in each inputs to make the prediction even if its prediciton is wrong. For instance, imagine we feed to the model a not traversable patch with an obstacle and the network label is as traversable. Obsliously, the output is wrong but two situation may happened that effect the degree of 'wrongness'. First, the model could just have ignored the obstcle and looked away, meaning if was not even able to understand it should have loook there. Second, the network could have correctly look at the obstacle but thouhgt that maybe the obstacle was not tall enough, showing a correct ability to find and use important features in the map. We showned that, even when the predictions are wrong, our model always look at the most important features of each input to determine its traversability.

We divided those inputs in four classes based on the model's performance: worst, best, false positive and folse negative. Then, we took twenty inputs from those sets and appplied Grad-CAM as texture on the 3D render to better visualize which region of the inputs caused the prediction.

0.1.1 Best

Best patches have few obstacles. We can obsverse two main clusters of images, flat and slopes. Interesting, when a surface has uneven ground near the left part, so close the the rear legs of the robot, the model is more interesting in those spots, 1a, 1c, 1d, 1e, 1p, 1q, 1r, 1s. This an expected behaviour since if there is an obstacle near the rear legs, then the robot will not be able to advance since it will be stuck from the beginning.

Moreover, in other patches 1b, 1i, 1j, 1t, the model's also looks ahead of the robot. In those situation the robot is able to properly move at the beginning so the network must evaluate the possibility of obstacles ahead. There are two oblious cases, 1i and 1t. The first one is a totally flat surface, so the model will look as far as possible to the robot's position to check if there are obstacles. Simarly, in the second, a surface with a bump in the hend, the network controls that spot. So, correctly, the network analysis the first region of the patch that may contain an untraversable obstacle.



(a) 0.20cm (b) 0.26cm (c) 0.30cm (d) 0.35cm (e) 0.38cm

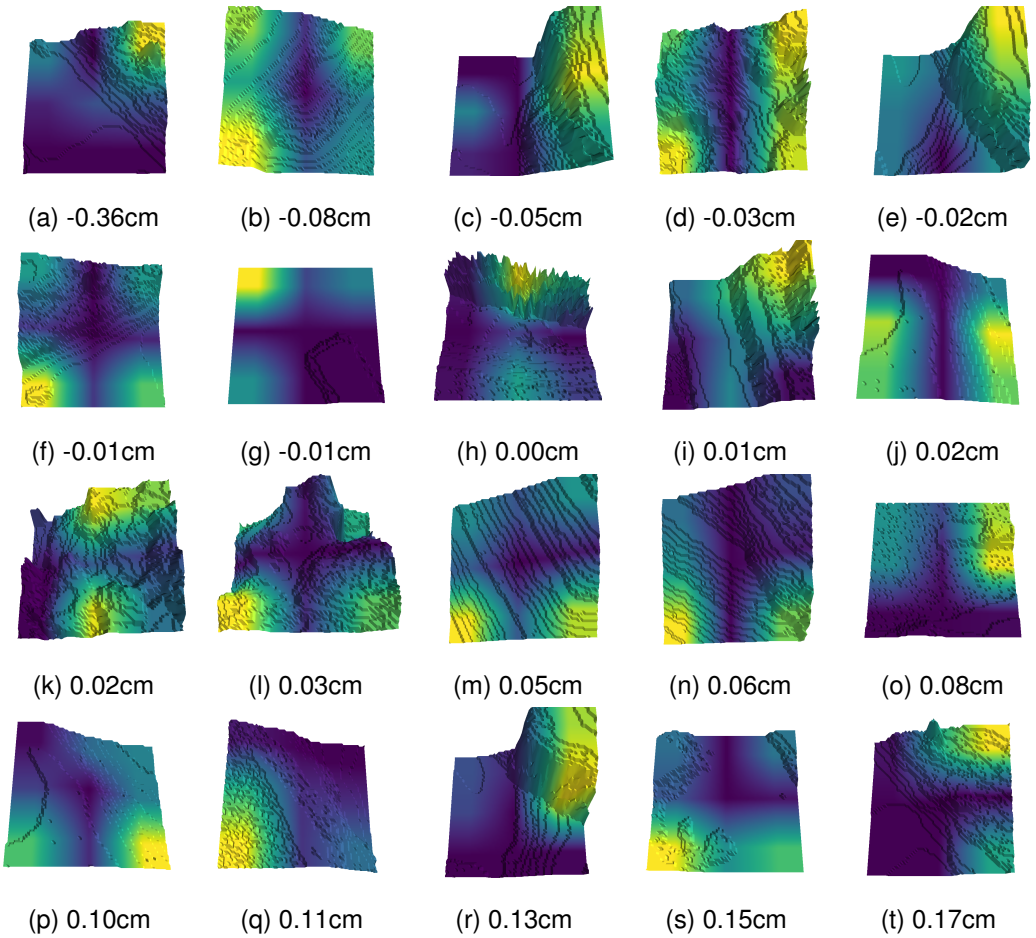
(f) 0.41cm (g) 0.44cm (h) 0.47cm (i) 0.49cm (j) 0.52cm

(k) 0.54cm (l) 0.57cm (m) 0.59cm (n) 0.60cm (o) 0.62cm

(p) 0.64cm (q) 0.66cm (r) 0.67cm (s) 0.68cm (t) 0.70cm

0.1.2 Worst

Those are the not traversable patches, we ordered by advancement. In this case, the region highlighted by the Grad-CAM are the features in the ground that contribute the most to make the patch not traversable. For instance, in some patches, 2b, 2f, 2t, the most not traversable features is directly under the robot legs, similar as before. While, for the others samples, 2a, 2c, 2d, 2e, ..., the attention of the network is on the obstacles in front of the robot. This is oblious in 2c, 2c, 2e, where the big wall on the right is almost totally highlighted. So, also in this case, the model is able to understand which part of the patch cause the prediciton.

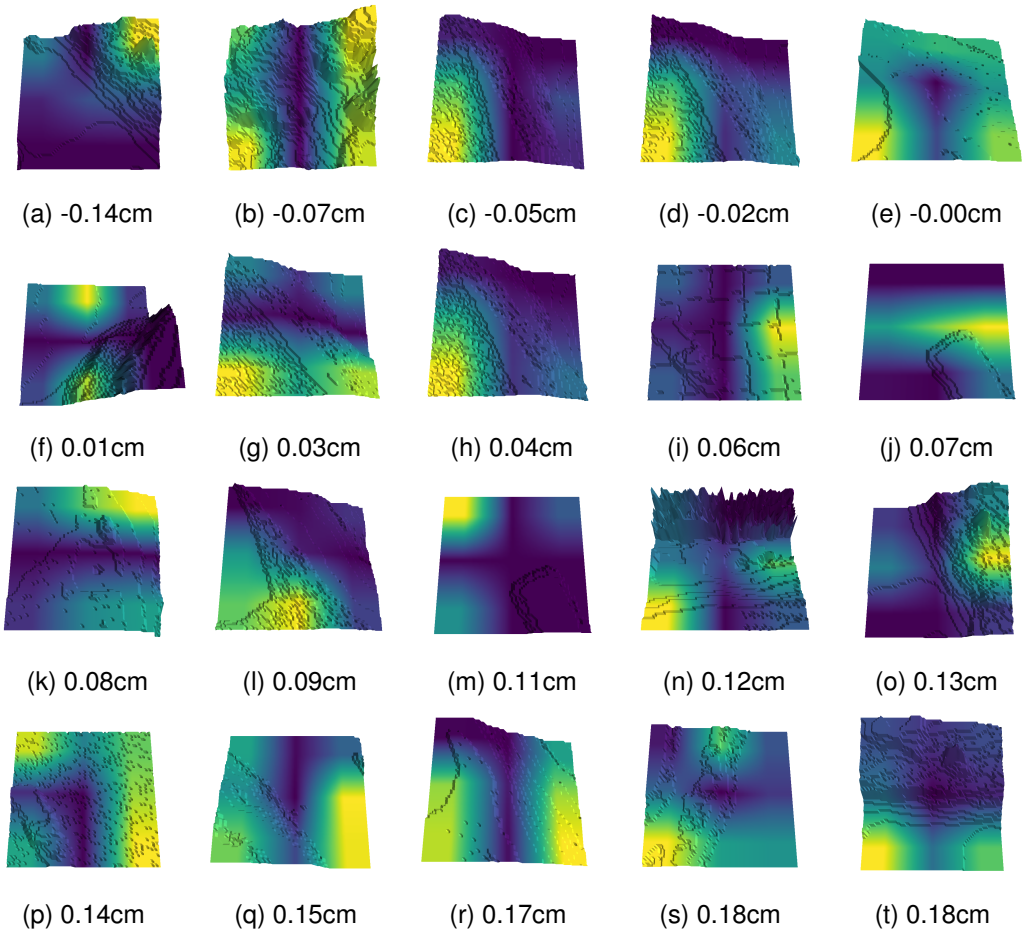


0.1.3 False Negative

Those are the inputs that were labeled as negative but predicted as positive. There are lots of interesting cases, we can cluster those patches in three groups: obstacles ahead, slopes and wall parallele to the robot. In figures 3a, 3f, 3n, the models is looking at the obstacle in front of the robot. Slopes, figures 3b, 3i, 3l, 3o, 3p, 3t, are not completely smooth, making harder to classify them. The last cluster of inputs are the ones with a wall parallel to the robot, figures 3n and 3t. In those cases the model looks at the initial position of the robot, left, and the final part of the patch. Intuitivelly, is trying to understand if the robot fits on the trail.

In general, most of the region of interested on those patches are located on the left, close to the position of the robot's leg. Correctly, even if the prediction is wrong, the model looks at the first region of the surface, located near the legs, that can cause non traversability. This shown a correct behaviour even with wrongly

prediciton meaning that the network is always looking in the correct spot.



(a) -0.14cm (b) -0.07cm (c) -0.05cm (d) -0.02cm (e) -0.00cm

(f) 0.01cm (g) 0.03cm (h) 0.04cm (i) 0.06cm (j) 0.07cm

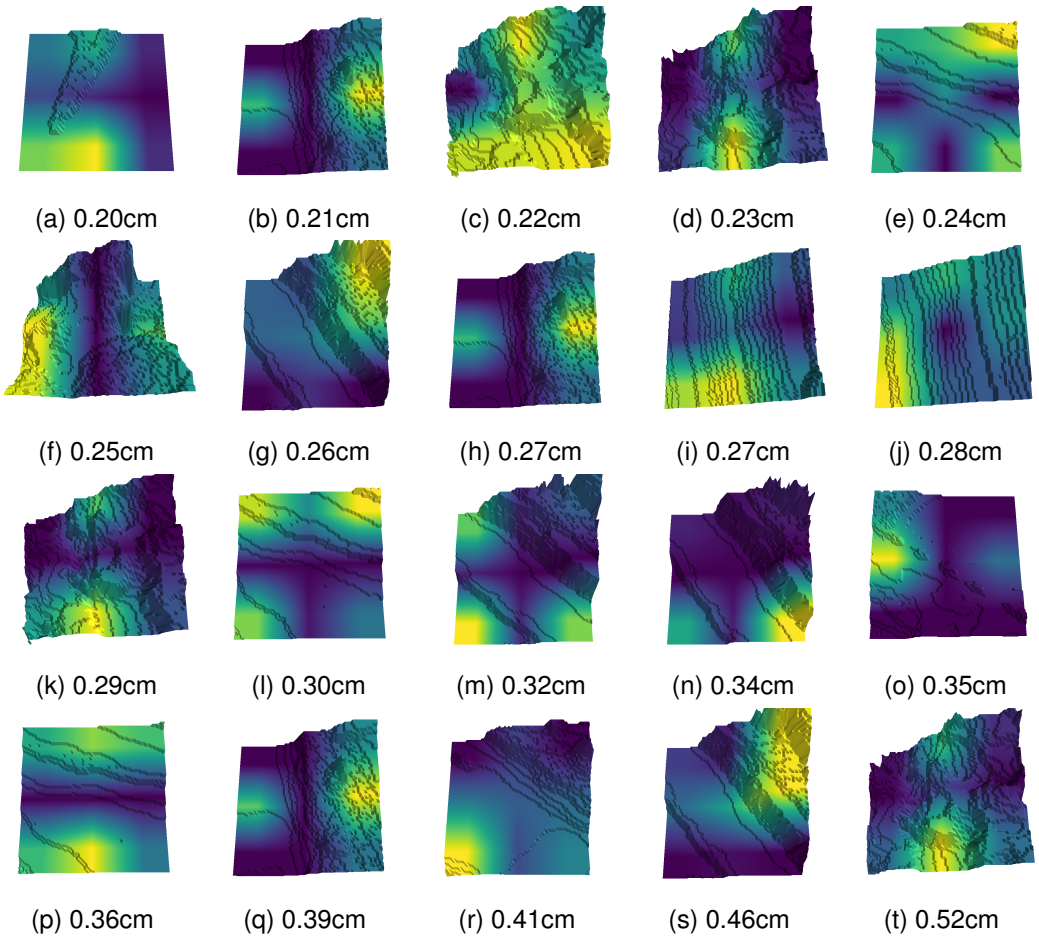
(k) 0.08cm (l) 0.09cm (m) 0.11cm (n) 0.12cm (o) 0.13cm

(p) 0.14cm (q) 0.15cm (r) 0.17cm (s) 0.18cm (t) 0.18cm

0.1.4 False Positive

Those are the most interesting ones, they are the samples that were labeled as traversable but predicted as not traversable. They includes different types of patches, obstacles ahead, slopes and flat regions. In each case the model looks at meaningfull features in the patches that can effect traversability. In the slopes, 4i and **??**, the first part of the surface is highlighted. Similar as before, the model is looking if the rear legs will be able to move and thought that those small steps will cause the robot not advance enough. The same consideration can be done for figure 4f. On the other hand, some inputs. definitely confused the model. For instance, in figures 4h and 4s the model is more interested in the big obstacle on the right part. This lead to classify those patches as not traversable, even if the obstacle is more distant than the treshold, in this case the network was totally confused. Interesting, figure 4n shows

an almost identical situation in which the network correctly looks at the initial part of the ground. We can deduce that even if the in almost any cases the network's prediction is caused by the correct part of the inputs, it can be confused by big obstacle near the end.



(a) 0.20cm (b) 0.21cm (c) 0.22cm (d) 0.23cm (e) 0.24cm

(f) 0.25cm (g) 0.26cm (h) 0.27cm (i) 0.27cm (j) 0.28cm

(k) 0.29cm (l) 0.30cm (m) 0.32cm (n) 0.34cm (o) 0.35cm

(p) 0.36cm (q) 0.39cm (r) 0.41cm (s) 0.46cm (t) 0.52cm