Cartesian-space to configuration-space estimation using a neural network

Sarthak Sachdeva
CS 5335, Northeastern University

Overview :

A lot of motion planning algorithms have been developed over the past few years. The transformation from the robot cartesian space (workspace) to the configuration space (C-space) has been critical. It involves converting the problem of collision-checking between n-dimensional objects to simpler, point-like, tests in the high-dimensional c-space. A machine needs to calculate all outcomes and plan a path that's collision free by calculating all points in the path while also exploring numerous failed routes. It's easier, although, for a human being to estimate a motion path by merely looking at the cartesian space. It is this motion planning by humans that enabled the exploration of mapping directly the cartesian space images to a configuration space. This was done by training a Neural network to approximate blocked vs free space categorization in n-dimensional c-space. (In our case, n = 2). Since the robot configuration is fixed, the way it interacts with an obstacle that's at a distance is similar for all objects at the same distance.

Setup :

In order to accomplish the goals for this project, an environment of a planar robot was set. This robot has two joints and two links of fixed width as shown below:

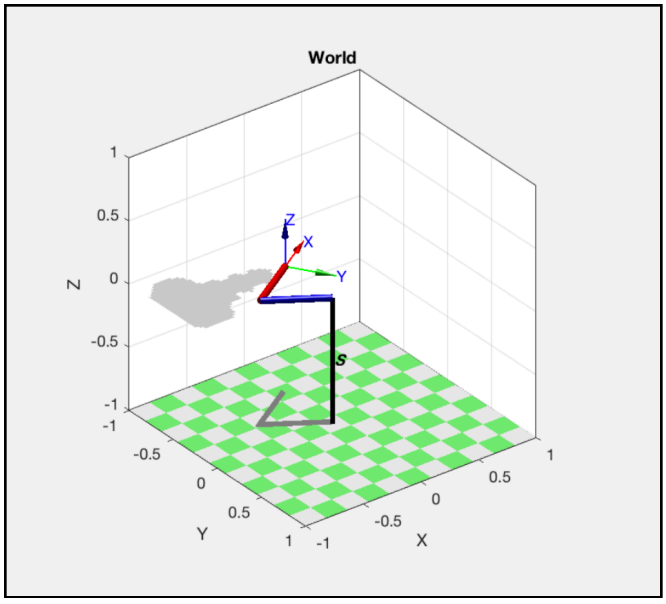


Figure 1: Environment containing a planar robot

The obstacles are planar and were random shapes fetched from an online dataset [1]. These shapes were then placed randomly around the robot in at most two quadrants. We then saved every experiment as an image with the robot at the center. The robot, however, was not printed to the image.

Next, we calculated the corresponding label for every image by sampling each joint angle about a 100 times while computing whether any part of our robot was colliding with an obstacle or not. Every free space was marked as zero (black) and the collision space was marked as one (white) as shown below :

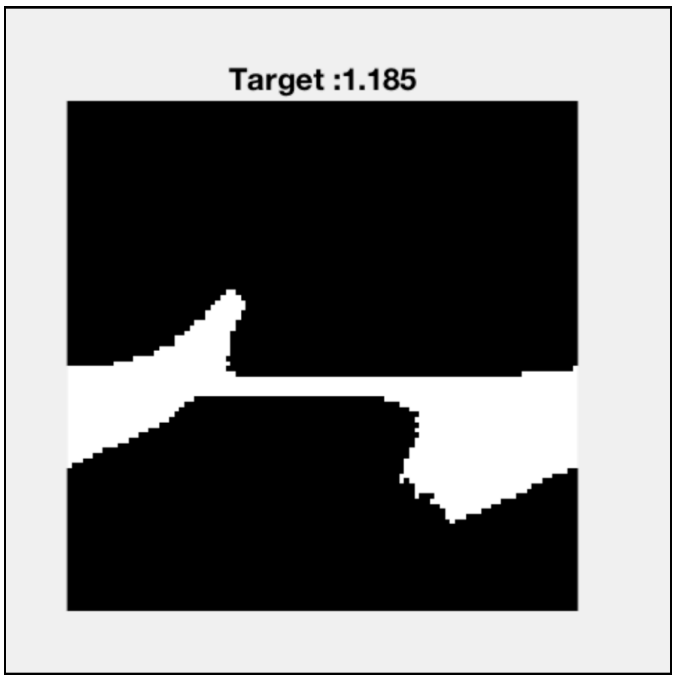


Figure 2 : Cartesian Space

Learning:

In order to accomplish the learning, we set up a neural network with pixels of image as our input and the pixels of the label image as output. Our starter model had just a one-layer architecture similar to a linear regression as shown in the figure underneath.

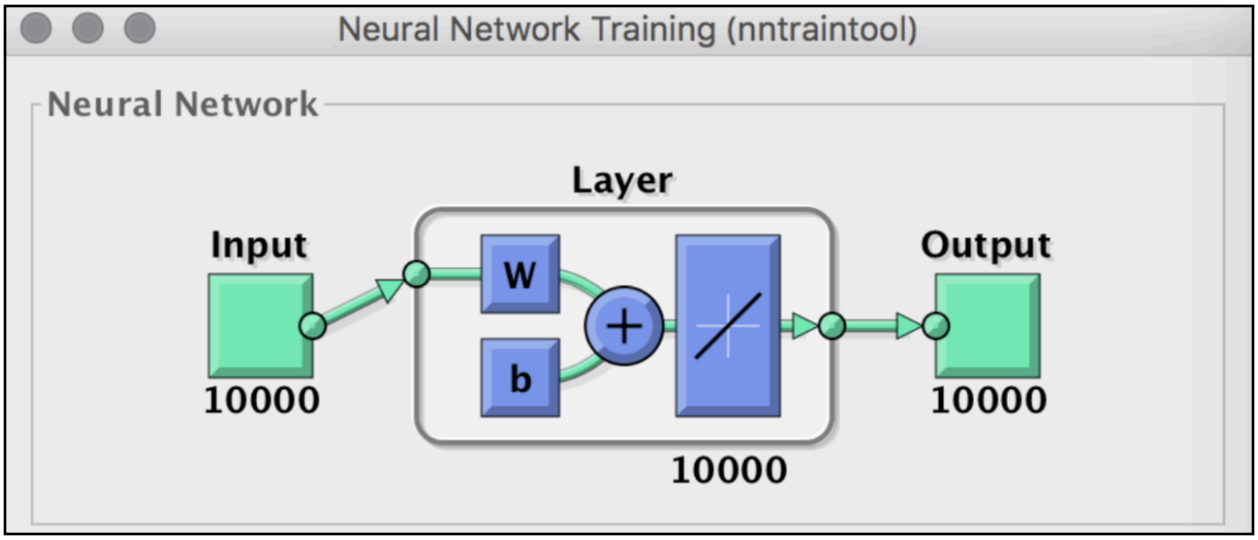


Figure 3 : Neural Network Training

We created various different resolutions of input and output images to compare model accuracy and performance. The resolutions were :

- 10x10
- 30x30
- 50x50
- 100x100

The 10x10 models were, naturally, faster for training and predicting with an average error rate of 0.39%. On the other hand, the 100x100 resolution models were much more accurate in predicting the cartesian-space. The average accuracy of the 100x100 model was 99.999%, which was extremely accurate. It is represented in the figure below:

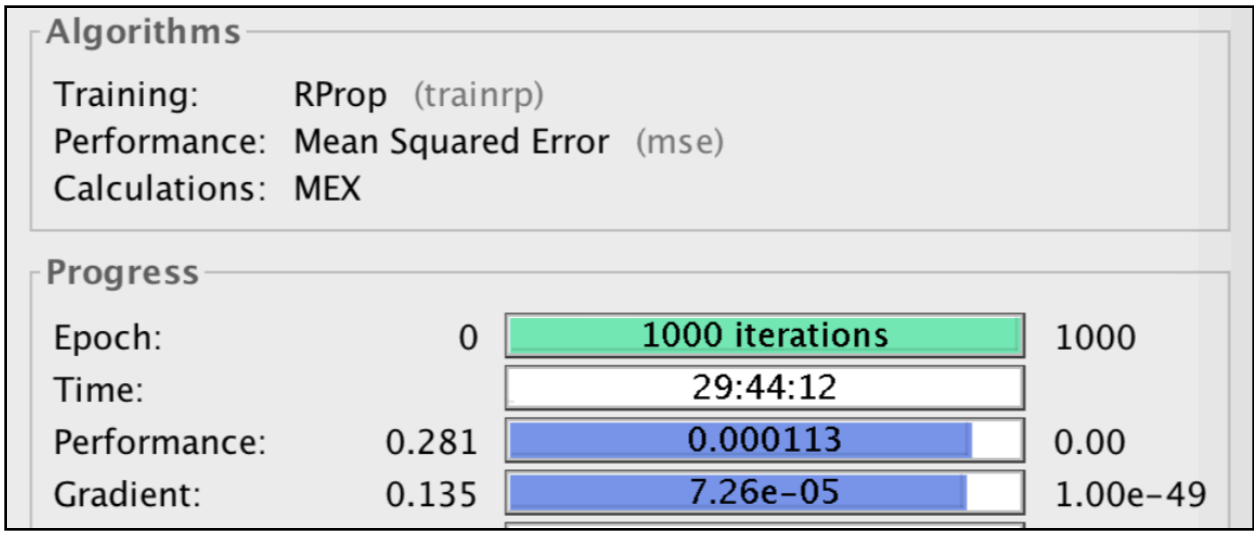


Figure 4 : Performance for the 100x100 model

Upon comparing performances across the different resolutions, we saw continuous improvement when we increased the resolution of the input and output image. Even though the NN shows stern penalties once you cross into higher resolutions, this could easily be improved using Auto Encoders. These encoders help reduce the dimensionality of the inputs reliably. The figure below represents these performance comparisons.

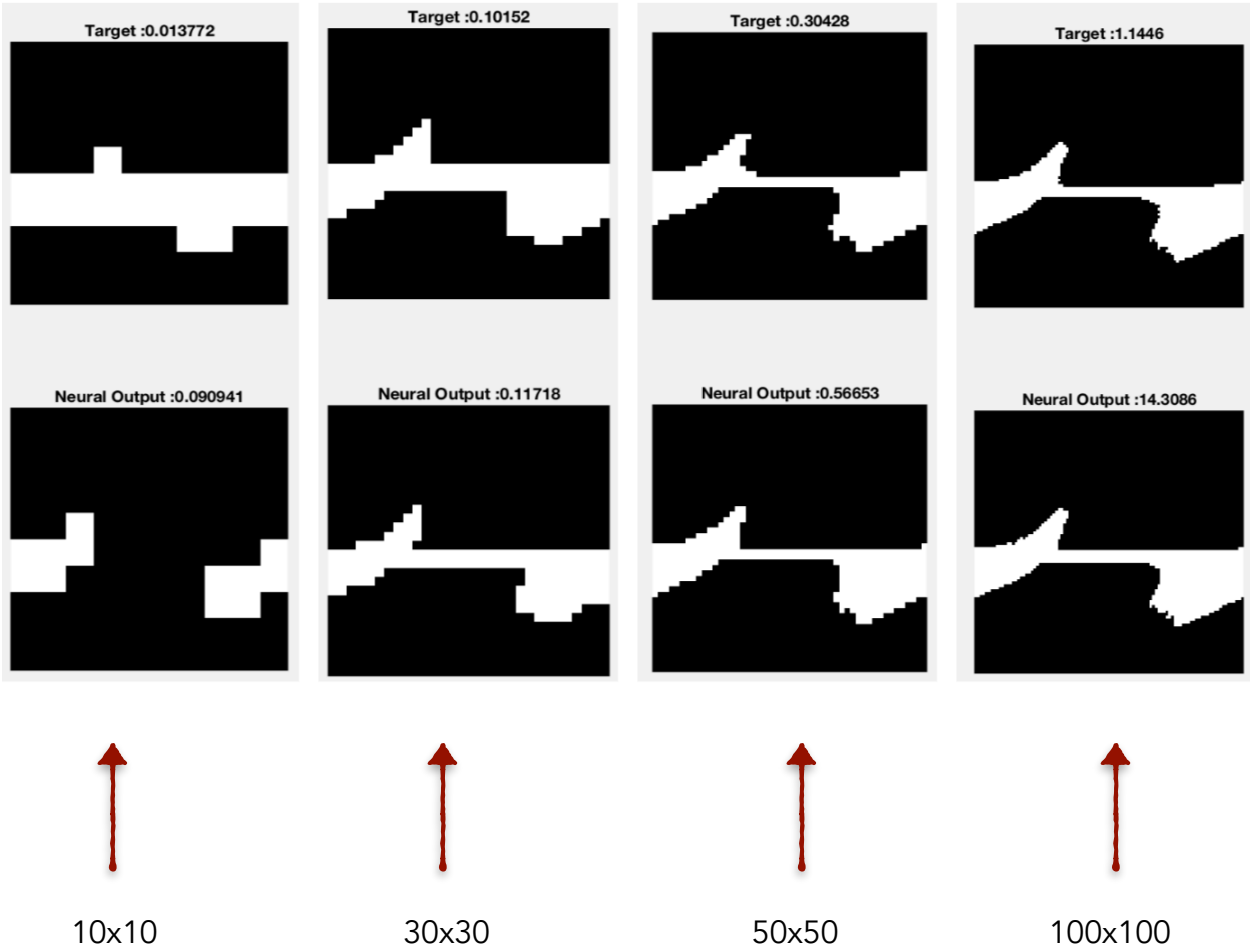
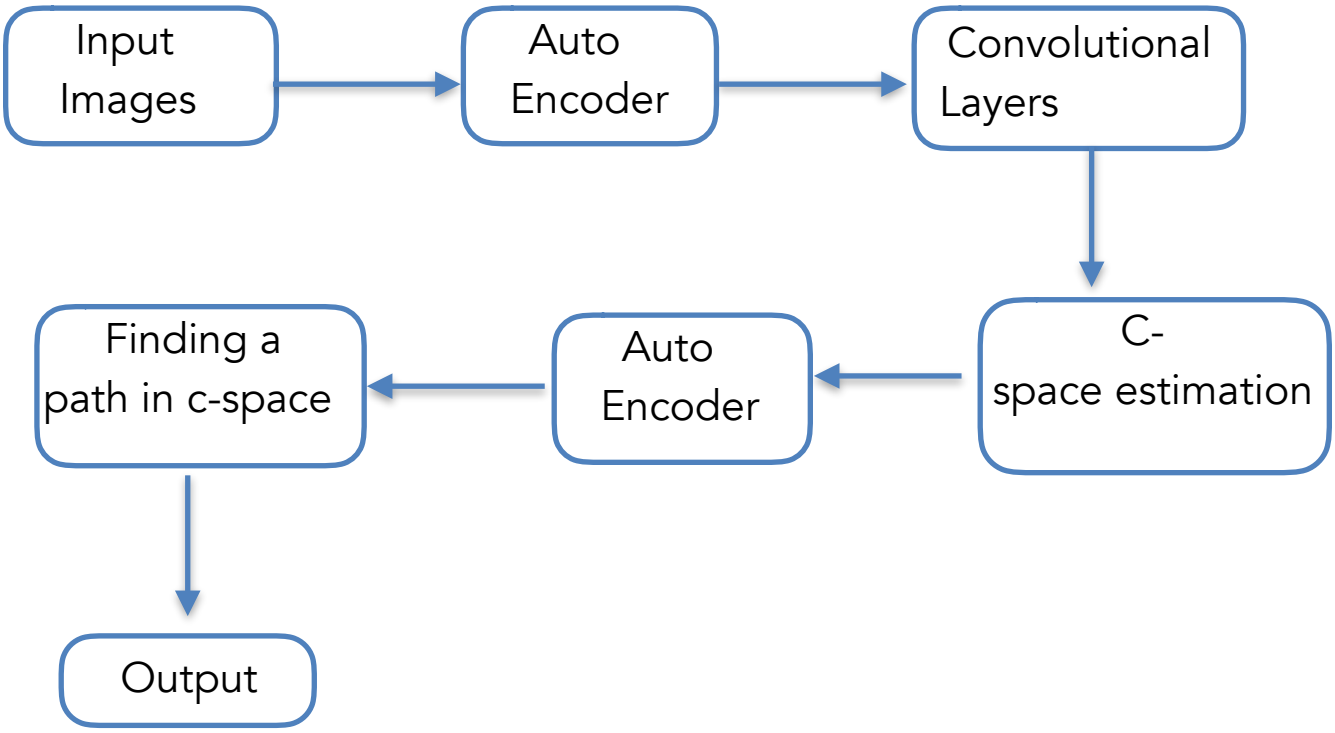


Figure 5 : Performance comparison for different resolutions (Numbers in the header labels for "Target" and "Neural output" represent the time taken for the calculation)

Future Scope:

This project could be expanded to train a robot on a full-scale with multiple joints to predict feasible joint configurations. This could make the path planning process more intuitive for each robot configuration. A design flow as follows could bolster this scope and help us get a performance balanced output:



Citations:

[1] “Shapes Dataset,"http://www.dabi.temple.edu/~shape/MPEG7/dataset.html.