Robot Learning of Social Path Planning Using a Simulator

Timo Mulder[a] Maarten van Someren[a]

[a] *University of Amsterdam, P.O. Box 94216 1090ED Amsterdam*

There exist several methods for a robot to learn how to navigate towards an assigned location in a socially acceptable manner. Most of these methods need to deal with the discretisation of a continuous space and the modelling of the dynamics of the world. To model these dynamics, the actions of the robot need to be discretised as well. In the current project, two infrequently described solutions to the above mentioned problems successfully build a basis for learning social navigation. These solutions involve a hexagonal grid and a probabilistic transition function. They are developed for the TERESA project, a project that constructs a socially intelligent semi-autonomous telepresence robot. Social navigation is part of the robot's social intelligence, which, in turn, the current project developed and tested a method for.

Various concerns made it convenient to discretise the environment using a hexagonal grid instead of a more typical square grid. Given that the robot can move in four directions, when the robot wants to move diagonally, it has to make a 'stair-like' movement, which may look unnatural and incompetent. This problem could be solved by giving the robot eight degrees of freedom on the grid. Nevertheless, the probability of ending up in the desired next state may be relatively small compared to horizontal and vertical movements. Therefore, the expectation is that the robot will never make use of the diagonal movements. In contrast, hexagons yield a consistent approximation of distances in a relatively wide range of directions. A reason for not choosing polygons containing more edges than a hexagon is that it would require a considerably greater amount of calculations for the transition function. Moreover, only triangles, squares and hexagons can be tiled in a regular manner.

To create a hexagonal grid, two square grids were created with dots as their centerpoints. Then, when Δ is the distance between two centerpoints in the x-direction, the offset for the centerpoints of the second grid relative to those of the first grid should be $\frac{\Delta}{2}$ and $\frac{\Delta}{2\sqrt{3}}$ in the x and y-direction, respectively. Together, these centerpoints form the centerpoints of the hexagonal grid. By taking the minimal Cartesian distance to those centerpoints, the borders of the hexagonal states can be drawn.

The discretised environment is in fact a continuous space. Therefore, when the robot's actions are controlled by a certain policy, the dynamics of the environment should be taken into account. Therefore, a transition function is necessary to model these dynamics. The transition function calculates probabilities of arriving in a certain state, given the current state and action. Those probabilities are obtained from recordings of driving random trajectories with the robot in a simulator. The actions are discretised in six directions by measuring the angle between two consecutive locations of the robot.

Since reaching all the states using every possible action (multiple times) is considerably time-consuming, the current project proposes a *template-style* approach for modelling the outcome of the transition function. This approach involves using relative state positions from only neighbouring states of the current state, rather than using absolute state positions. Also, the total number of times that a particular relative next state was reached by moving in a specific direction is averaged over the action's total, resulting in a probability distribution. As a consequence, there is a significant reduction in the amount of data that is needed to model the dynamics. The *template-style* approach also yields time-invariance, since the probability of arriving in a particular next state is only dependent on the current state. This approach made it possible to acquire a saturated transition model for a total of 128400 states, using less than 445 seconds of recorded data, containing 1241 state changes. Whether the model was saturated or not was measured by comparing the transition model of the whole dataset with models obtained using smaller subsets of the same data. The model did not notably change between the whole dataset and one containing only 81% of the data; hence, the model was saturated. A policy for the robot's behaviour was created by running an inverse reinforcement learning algorithm on the combination of expert examples and the transition model. High similarity between the resulting policy and the recorded samples confirm proper working of the obtained model (average -$\log$-likelihood of 0.097).