Introduction

Objective: Create a controller to generate failure trajectories when a mobile robot fails to reach a waypoint. Additionally, make a predictor that identifies the distribution of next locations the robot could take given its state.

Motivation
- Robots navigating amongst crowds of people need to obey implicit human social rules and avoid many dynamic obstacles.
- For long term planning to distant goals, we seek to separate planning from controllers to handle small steps.
- Because these controllers may fail, we seek to develop a "failure controller" to guide the robot along a contingency trajectory.
- We would also like to predict this trajectory at planning time so the planner can perform state expansions from it.

Approach

Failure Controller
- Model navigation as a Markov Decision Process \((S, A, R, P, \gamma)\) to achieve the maximum long term return.
- Use deep reinforcement learning (specifically the DQN algorithm [1]) to learn a function \(Q(s_t, a_t)\) which gives the long term reward of action \(a_t\) from state \(s_t\).
- Once we know \(Q\), we can easily construct an optimal trajectory by always selecting action \(\arg\max_a Q(s_t, a)\).

Failure State Prediction
- Simulate and record trajectories for the failure controller.
- Train a neural network to predict the distribution of next locations given the current state. We assume that the final distribution is Gaussian, so the network learns the parameters \(\mu, \sigma\), \(\mu, \sigma\), \(\sigma\), \(\rho\) that characterize it.

Failure Controller Generates Safe and Predictable Trajectories

A comparison between our controller (left), a social forces controller (middle), and a do nothing controller (right). The initial and final locations of the robot and each human are shown, connected by a dotted line to illustrate their trajectories. The social forces trajectories continue to the left in an approximately straight line until they reach their time-out.

Failure Predictor

Examples of the predictor’s output. At each time step, the failure predictor estimates the robot’s next position and its uncertainty in that prediction. The dashed green ellipse shows the region in which the predictor is 99% confident the robot will be next and the black square shows the robot’s actual next location. The top row shows results from training data which have much lower variance as expected.

Failure Controller Balances Safety and Efficiency

<table>
<thead>
<tr>
<th></th>
<th>RL (Ours)</th>
<th>Social Forces</th>
<th>Do Nothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path Length</td>
<td>1.23 (0.74)</td>
<td>14.70 (0.00)</td>
<td>0.07 (0.12)</td>
</tr>
<tr>
<td>Angular Distance</td>
<td>21.15 (13.27)</td>
<td>4.18 (1.47)</td>
<td>10.28 (6.95)</td>
</tr>
<tr>
<td>Time</td>
<td>5.61 (2.96)</td>
<td>10.00 (0.00)</td>
<td>3.39 (3.45)</td>
</tr>
<tr>
<td>Collisions</td>
<td>0.02 (0.14)</td>
<td>0.00 (0.00)</td>
<td>0.75 (4.74)</td>
</tr>
<tr>
<td>Intrusions</td>
<td>34.72 (47.11)</td>
<td>0.76 (2.41)</td>
<td>190.58 (60.71)</td>
</tr>
</tbody>
</table>

Comparison between the presented controller, a social forces controller, and a do nothing controller. The mean of each metric over 100 runs is shown with the standard deviation in parentheses.

Conclusion and Future Work

This work presents a failure controller which successfully balances safety with path efficiency and a predictor to forecast its trajectories at planning time.

Future Work
- Apply this method to other controllers for different behaviors (for example, overtaking pedestrians).
- Develop a robust way to determine, at execution time, when to end the failure controller.

References


Acknowledgements

- Thank you to Rachel Burcin and Professor John Dolan.