Adaptive Navigation For Autonomous Robots

Authors: Matt Knudson and Kagan Tumer
Oregon State University

Presented by: Raj Dasgupta
Overview

- **Main objective:** Robot navigation in partially observable and dynamic environments that are stochastic in nature

- **Using domain knowledge to navigate**
  - Requires high sensor usage, computation and power requirements – infeasible or impossible in certain domains

- **Map from sensor inputs to actions**
  - Statistically capture key behavioral objectives
  - No environment model or domain knowledge requires to be built
Proposed Approach and Contributions

- Neuro-evolutionary based navigation
  - Policy search methods
  - Computationally inexpensive (e.g., neural network), already shown to be successful in many applications

- Contributions
  - State and action space that are robust to sensor/actuator noise
    - Ranking (quality) between 360 possible paths from current location towards goal
  - Neural network to evolve parameters that assign quality to possible paths
  - Validation on physical robots
Related Work

- Heuristic search approaches: A*, D*, variants
- Navigation techniques to account for uncertainty in perceived sensor data and its interpretation (GoldbergMataric; WhitsonStone et al.)
- Navigation techniques to account for incorrectly generated environment models (don’t retain model or adapt to changes to environment) (ClementDurfee; FoxBurgardThrun)
- Reinforcement learning for navigation
  - ErdenLeblebicigolu – gait generation for six–legged robot
  - El–FakdiCarrerasPalomera – underwater vehicle control
Robot Navigation

- Limited sensing possible by robot
- Non-deterministic actions
- Two types of navigation
  - Rule-based: Probabilistically select a rule to navigate from a rule-base
  - Neuro-evolved:
    - Evaluate environment, determine 360 paths (one for each degree)
    - Determine quality metric to each path – evolve quality metric using multi-layer feed-forward neural net
State and Action Spaces

- **Incoming state variables**
  - **Object Distance:** one distance value along each of 360 degrees at robot’s current position
  - **Destination Heading:** Angular difference between current heading and goal heading of robot

- **$X(\theta_i)$:** represents the quality of path in direction $\theta_i$
Rule-based Navigation

- Behavior (movement direction) of robot specified by a rule, which does not change

Parameters used by rule-based navigation

- $T$ Indexes episodes
- $t$ Indexes time-steps within each episode
- $\theta_i$ Angle of potential path $i$
- $\alpha_{des}$ Vehicle relative angle to destination
- $P_{direct}$ Probability that the destination
- $P_{safe}$ Probability that the potential
- $X(\theta_i)$ Path quality assigned to
- $\alpha_u$ Chosen vehicle relative ro
- $F(X(\alpha_u))$ Linear mapping of p
- $V_u$ Chosen robot speed.

- Sudden changes in goal location between two time steps does not require calculating complex multi-step paths every time step

Rule-based navigation algorithm

For $T < T_{max}$ Loop:

For $t < t_{epi}$ Loop:

Capture current state $s$

For $\theta_i \leq 360$ Loop:

1. Calculate $P_{direct}$ given $\theta_i$ and $\alpha_{des}$
2. Calculate $P_{safe}$ given $d_{\theta_i}$
3. $X(\theta_i) \leftarrow P_{direct} \times P_{safe}$
4. $\alpha_u \leftarrow \text{argmax} X(\theta_i)$
5. $V_u \leftarrow F(X(\alpha_u))$
Neuro-Evolution

- Same state variables (distance to obstacle, angular difference from goal) and actions (quality of paths along each one of 360 heading angles from current position)
- Instead of using probability of safe/direct path uses artificial neural network (ANN) to calculate path qualities
- One ANN selected from 20 neural nets using evolutionary algorithms with domain specific objective function
- Selected ANN run to determine path quality for every path at every time step

```
Initialize N networks at T = 0
For T < T_{max} Loop:
  1. Pick a random network N_i from population
     With probability \( \epsilon \): \( N_{current} \leftarrow N_i \)
     With probability \( 1 - \epsilon \): \( N_{current} \leftarrow N_{best} \)
  2. Mutate \( N_{current} \) to produce \( N' \)
  3. Control robot with \( N' \) for next episode
     For \( t < t_{epi} \) Loop:
       3.1 For \( \theta_i \leq 360 \) Loop:
          Run \( N' \) to produce \( X(\theta_i) \)
          \( \alpha_u \leftarrow \arg\max_X X(\theta_i) \)
          \( V_u \leftarrow F(X(\alpha_u)) \)
       3.2 \( \alpha_u \leftarrow \arg\max_X X(\theta_i) \)
       3.3 \( V_u \leftarrow F(X(\alpha_u)) \)
  4. Rank \( N' \) based on performance (objective function)
  5. Replace \( N_{worst} \) with \( N' \)
```
Robot Capabilities

- Orientation: Inertial sensing
  - Initial orientation needs to be known
- Environment sensing: 8 sonars in 360 degree ring around robot; each sonar has 45 degree cone; range 4m
- Actuation: Differential drive, two-wheels, one caster wheel
Objective Function

- Three parameters
  - Total distance traveled to reach goal
  - Total time taken to reach goal
  - Time take to recover from collisions with obstacles

\[ R(s) = \alpha(d_{\text{best}} - d_{\text{actual}}) + \beta(t_{\text{best}} - t_{\text{actual}}) - \gamma t_{\text{collision}} \]  \hfill (1)

- \( \alpha = 1.0; \beta = 10.0; \gamma = 10.0 \)

Rule-based     Neuro-evolutionary
Simulation Experiments

- Environments
  - No obstacles
  - Fence obstacle: First openspace; then fence (handle emergent situation during navigation)
  - Dense environment: Many, randomly scattered obstacles (handle complex environment rich with sensor information)
  - Dense environment with randomly generated sensor and actuator noise (handle robustness to real-world situational signal noise)
No Obstacles Experiments

- Rule-based performs best; because there are no obstacles, no adaptation in path is required
- Adaptive navigation includes $\varepsilon$-exploration, therefore its mean is below rule-based

Fig. 6. Left: The experimental arena for the No Obstacles situation. The figure is to scale and shows a sample robot, the destination (diamond) as well as the possible starting locations for an episode (inner rectangular area). The number of possible initial conditions as well as the approximate percentage of those being direct paths are listed at the base of the figure. Right: The results of the learning for the No Obstacles situation. The objective function is plotted for the random, rule-based, and neuro-evolutionary algorithms as an average over 40 iterations.
• Rule-based still performs best; because only little adaptation in path is required
• Neuro-evolutionary navigation converges to 50% of rule-based
• Neuro-evolutionary performs better as sonar information (from obstacles) allows to ANN learn movements around obstacles more (quickly)
Dense Environment Paths

- Rule-based prefers safety
  - Heads to midpoint between two obstacles
- Neuro-evolutionary prefers distance (directness)
  - Goes as close to obstacle as safely possible

Fig. 9. Sample robot paths are displayed. The neuro-evolutionary path is generated using data produced during runs with converged networks. Initial placement and heading is randomized at every iteration, so paths were selected that began in approximately the same location. Circles are obstacles, and robot orientation on each path indicate the robot’s initial heading.
Rule-based performs more poorly with more obstacles

Neuro-evolutionary starts losing performance after 15 obstacles

Too much clutter from 20 obstacles

Neuro/random algorithms unable to find path to goal
Dense Environment with Noise

- 5% random Gaussian Noise added initially;
  - Increased to 10% around 1000th episode
- Neuro-ev. approach converges slower than no-noise setting

Presence of noise does not affect neuro-ev. approach performance
- Once good behaviors are learned they remain in system
Effect of Amount of Noise

Fig. 14. The impact of signal noise is shown. Maximum system performance achieved during a 2000 episode experiment, averaged over 40 statistical runs, is plotted against varying noise levels.
Physical Robot Experiments

- Pioneer 3DX robot
- Rule-based approach required parameter tuning
- Neuro-evolutionary approach
  - Required no parameter tuning
  - Least times to handle collision

![Figure showing the environment used for functional experiments. The figure is to-scale and represents the Autonomous Systems research lab. Obstacles are solid black squares and are 14" on each side. Also shown are the 5 sampled starting locations and the destination.](image)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule-based (sim)</td>
<td>-8.8 ± 0.3</td>
<td>-11.4 ± 1.0</td>
<td>-9.5 ± 0.4</td>
<td>-8.1 ± 0.1</td>
<td>-6.5 ± 0.2</td>
</tr>
<tr>
<td>Rule-based (real)</td>
<td>-8.2 ± 0.4</td>
<td>-8.1 ± 0.3</td>
<td>-8.3 ± 0.3</td>
<td>-8.6 ± 0.3</td>
<td>-7.1 ± 0.4</td>
</tr>
<tr>
<td>Neuro-evolution (sim)</td>
<td>-12.9 ± 2.7</td>
<td>-9.1 ± 2.5</td>
<td>-6.4 ± 0.2</td>
<td>-6.2 ± 0.2</td>
<td>-3.5 ± 0.2</td>
</tr>
<tr>
<td>Neuro-evolution (real)</td>
<td>-5.8 ± 0.5</td>
<td>-6.9 ± 1.1</td>
<td>-5.4 ± 0.2</td>
<td>-6.2 ± 0.3</td>
<td>-5.9 ± 0.8</td>
</tr>
</tbody>
</table>

Table 1: The results of experimentation for both navigation algorithms. The numbers shown are the same system evaluation $R(s)$ discussed in Section 3.2 and shown in Section 4. A, B, C, D, and E represent different starting positions, illustrated in Fig. 15.
Conclusions

- Navigation approach for robots that have
  - Limited sensing
  - Cannot build and store map of environment
- Neuro–evolutionary approach better than probabilistic distributions in more complex scenarios
  - But breaks down in very complex environments
    - Due to limited sensing, no model–building