Cognitive Agents for Autonomous Robots

University of Dayton

Air Force Research Laboratory
Team Members

Air Force Research Laboratory
- Kerry Hill - Sensors Directorate
- Todd Jenkins - Sensors Directorate
- Scott Douglass - Human Effectiveness Directorate
- Trevor Bihl - Sensors Directorate
- Ashley DeMange - Sensors Directorate

University of Dayton
- Tarek Taha
- Chris Yakopcic
- David Fan (UDRI)
- Nayim Rahman
- Tanvir Atahary
- Alex Beigh (UDRI)
INRC Project Overview

This project will develop cognitive agents for an autonomous robot. The system will take inputs from sensors (an Inilabs spiking camera and other sensors) and process these as inputs to a pair of cognitive agents:

- A high-level decision agent that takes a set of mission requirements, such as what order to complete a set of objectives based on the priorities for each objective.
- A motor control agent that accomplishes a specific set of physical tasks as directed by the high-level agent.
Progress: Reactive Robot (no CECEP or spiking camera)

- Begins with perception
  CNN detections from video camera sent to Reasoner
- Reasoner then decides the platform’s course of action
- Action sent to the navigation and control components
- Interaction with world (real or M&S) sensed, movement triggers changes in its perception and the loop repeats.
Example Results

High level intentions:
- “Feed”
- “Flee”
- “Explore”

General “Moods” for the reasoned:
- “Hunger”
- “Fear”
- “Curiosity”

Example output from perceptor, with example object classes. Reasoner gives intents, which result in control outputs and moods (which are fed back to the Reasoner).

Bihl, T. J., Jenkins, T. R., Cox, C., DeMange, A., Hill, K., Zelnio, E. (2019) “From the lab to the internship and back again: Learning autonomous systems through creating a research and development ecosystem,” AAAI/EAAI, Honolulu, HI
High Speed Cognitive Domain Ontologies Using Loihi Spiking Neurons

- CECEP Architecture
- The CDO is the decision making engine within the CECEP architecture
- These very simple examples quickly become very complex in realistic systems
  - Need to find best of billions of solutions
Objective

- Optimized resource allocation is extremely computationally expensive
- We need low SWaP alternatives, large problems are currently prohibitively expensive to solve.
- This is done using a series of spiking neurons that fire according to the most logical vehicle assignment options
- This work covers a MATLAB implementation of the spiking neuron based algorithm

<table>
<thead>
<tr>
<th>Allocation Problem Size</th>
<th>Number of Possible Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 × 2</td>
<td>9</td>
</tr>
<tr>
<td>4 × 4</td>
<td>625</td>
</tr>
<tr>
<td>6 × 6</td>
<td>117,649</td>
</tr>
<tr>
<td>8 × 8</td>
<td>43,046,721</td>
</tr>
<tr>
<td>10 × 10</td>
<td>25,937,424,601</td>
</tr>
</tbody>
</table>
Objective

- Optimized resource allocation is extremely computationally expensive
- We need low SWaP alternatives, large problems are currently prohibitively expensive to solve.
- This is done using a series of spiking neurons that fire according to the most logical vehicle assignment options
- This work covers a MATLAB implementation of the spiking neuron based algorithm

**Vehicle 1 Goes to Target 1**  
**Vehicle 2 Goes to Target 3**  
**Vehicle 3 Goes to Target 4**  
**Vehicle 4 Goes to Target 3**  
**Vehicle 5 Goes to Target 5**  
**Vehicle 6 Goes to Target 8**  
**Vehicle 7 Goes to Target 5**  
**Vehicle 8 Goes to Target 7**
**Simplified Algorithm Block Diagram**

- Single neuron holds connection between one vehicle and one target
  - Capable of allocating N vehicles for M separate targets using N×M neurons
  - Spike accumulation is proportional to a weighted sum of priority, success, and time

- Weight Parameters
  - TTA: Time to the target
  - Priority: Necessity of reaching target
  - TOT: Hold time for vehicle once target is reached
  - Probability of Success: Likelihood that a target will be completed by a certain vehicle
  - TTC = TOT + TTA

- Control Parameters
  - CM: Connectivity matrix hold vehicle-target compatibility
  - $\tau$: A vehicle can only be assigned to one target
  - $\beta$: Penalize (but do not stop) multiple vehicles from reaching a single target
• Single neuron holds connection between one vehicle and one target
  • Capable of allocating N vehicles for M separate targets using N×M neurons
  • Spike accumulation is proportional to a weighted sum of priority, success, and time

• Weight Parameters
  • TTA: Time to the target
  • Priority: Necessity of reaching target
  • TOT: Hold time for vehicle once target is reached
  • Probability of Success: Likelihood that a target will be completed by a certain vehicle
  • TTC = TOT + TTA

• Control Parameters
  • CM: Connectivity matrix hold vehicle-target compatibility
  • τ: A vehicle can only be assigned to one target
  • β: Penalize (but do not stop) multiple vehicles from reaching a single target
Example Allocation

Connectivity Matrix Response

Control Variables

Constraint Handler

Vehicle Control ($\tau_i$)

Target Control ($\beta_j$)

Connectivity Matrix Response

Vehicle 1

Target 1

Target 2

Target 3

Target 4

Target 5

Vehicle 2

Vehicle 3

Vehicle 4

Vehicle 5

Target Control ($\beta_j$)

Connectivity Matrix Response

Vehicle Control ($\tau_i$)

Spiking Output

Resulting Allocation

Bias Input

Spike Layer
Example Allocation

Vehicle Control Response

Vehicle Control Response

Control Variables

Connectivity Matrix \( (C_{m_{ij}}) \)

Spike Inputs \( v_{11}, \ldots, v_{1M} \)

\( \ldots \)

\( \ldots \)

\( \ldots \)

Vehicle Control \( (\tau_i) \)

Target Control \( (\beta_j) \)

Spiking Output

Resulting Allocation

Vehicle 1

Target 1

Target 2

Target 3

Target 4

Target 5

Vehicle 2

Vehicle 3

Vehicle 4

Vehicle 5
Example Allocation

Target Control Response

Target 1 Target 2 Target 3 Target 4 Target 5
Vehicle 1 20 20 20 20 20
Vehicle 2 20 20 20 20 20
Vehicle 3 20 20 20 20 20
Vehicle 4 20 20 20 20 20
Vehicle 5 20 20 20 20 20

Target Control Response

Constraint Handler
Vehicle Control ($\tau_i$)
Target Control ($\beta_j$)
Connectivity Matrix ($C_{m_{ij}}$)
Isolated Neuron Grid

Control Variables
Bias Input
Spike Layer
Spiking Output
Resulting Allocation

Spike Inputs:

Resulting Allocation:
Algorithm Comparison

<table>
<thead>
<tr>
<th>Allocation Size</th>
<th>Exhaustive Search Using CPU/GPU</th>
<th>Spiking System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline CDO Reward</td>
<td>Baseline CDO Result</td>
</tr>
<tr>
<td>3x3</td>
<td>18.8703</td>
<td>[2 1 1]</td>
</tr>
<tr>
<td>4x4</td>
<td>11.3777</td>
<td>[4 1 1 3]</td>
</tr>
<tr>
<td>5x5</td>
<td>22.6219</td>
<td>[1 5 2 4 1]</td>
</tr>
<tr>
<td>7x7</td>
<td>48.448</td>
<td>[4 2 2 6 5 6 7]</td>
</tr>
<tr>
<td>8x8</td>
<td>40.8782</td>
<td>[1 3 4 7 5 3 6 8]</td>
</tr>
</tbody>
</table>

- The best allocation for each case was determined using a GPU exhaustive search.
- The table compares this result to the approximate result obtained from the Loihi spiking system.
Timing Comparison

- Runtime comparison between the exhaustive search and spiking system

<table>
<thead>
<tr>
<th>Allocation Size</th>
<th>CDO Search Time (CPU)</th>
<th>CDO Search Time (GPU)</th>
<th>Spiking System Time</th>
<th>Spiking System Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>2×2</td>
<td>2.6 ms</td>
<td>-</td>
<td>30.6 ms</td>
<td>0.08×</td>
</tr>
<tr>
<td>3×3</td>
<td>8.9 ms</td>
<td>-</td>
<td>31.2 ms</td>
<td>0.29×</td>
</tr>
<tr>
<td>4×4</td>
<td>84.7 ms</td>
<td>-</td>
<td>35.9 ms</td>
<td>2.36×</td>
</tr>
<tr>
<td>5×5</td>
<td>13,041 ms</td>
<td>-</td>
<td>42.2 ms</td>
<td>309×</td>
</tr>
<tr>
<td>6×6</td>
<td>-</td>
<td>223.6 ms</td>
<td>53.1 ms</td>
<td>4.21×</td>
</tr>
<tr>
<td>7×7</td>
<td>-</td>
<td>454.8 ms</td>
<td>53.9 ms</td>
<td>8.43×</td>
</tr>
<tr>
<td>8×8</td>
<td>-</td>
<td>480,205 ms</td>
<td>62.4 ms</td>
<td>7696×</td>
</tr>
</tbody>
</table>

- Solution space on an asset allocation problem grows with problem size
  - Neuron utilization grows at a much smaller rate

<table>
<thead>
<tr>
<th>Allocation Problem Size</th>
<th>Number of Possible Solutions</th>
<th>Number of Neurons</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 × 2</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>4 × 4</td>
<td>625</td>
<td>16</td>
</tr>
<tr>
<td>6 × 6</td>
<td>117,649</td>
<td>36</td>
</tr>
<tr>
<td>8 × 8</td>
<td>43,046,721</td>
<td>64</td>
</tr>
<tr>
<td>10 × 10</td>
<td>25,937,424,601</td>
<td>100</td>
</tr>
</tbody>
</table>
Next Steps

• Asset Allocation
  • More complete algorithm comparison
  • Study of maximum scalability
  • Results Found in:

• Loihi
  • Constraint Solving
  • Energy / Timing Benchmark