Automatic Extraction of Command Hierarchies for Adaptive Brain-Robot Interfacing

Matthew Bryan*, Griffin Nicoll*, Vibinash Thomas*, Mike Chung, Joshua R. Smith, Rajesh P. N. Rao

Computer Science Department, University of Washington, Seattle
*Undergraduate Student Researcher
This work was produced by undergraduate UW engineering students:

Matthew J. Bryan  Griffin Nicoll  Vibinash Thomas

And advised by:

Joshua Smith  Rajesh P.N. Rao

Presented by: Matthew Bryan
Acknowledgements

Center for Sensorimotor Neural Engineering
Brain-Computer Interfaces (BCIs) for severely paralyzed or “locked-in” persons.

- Artificial neural pathway
- Can we provide more independence via a BCI and a robotic proxy?
Non-invasive EEG

- Practical, but suffers from low throughput (< 60 bits/min)
- Fine-grained control impractical in long term
- But need flexibility to deal with a wide variety of situations
Hierarchical skill learning example:

- Learn individual pen strokes
- Learn to put pen strokes together to form letters
- Learn to put letters together to form words
- Etc.
Hierarchical BCI (HBCI):

- User performs tasks with lower-level skills
- HBCI observes user to learn higher-level skills
- User can execute higher-level skills directly
- Raises effective throughput of the interface
- Can work independently of the choice of BCI paradigm
System Overview

User

Brain signal

BCI

Robot State
Available Skills

Robot State

Available Commands

GUI

Robot

HBCI Control Agent

Robot Commands

UI Commands
System Overview

User -> Brain signal -> BCI

Robot State: Available Skills

GUI

Available Commands

HBCI Control Agent

Robot Commands

Robot

Robot State

UW Neural Systems Lab
UW Sensor Systems Lab
HBCI Control Agent:

- Extracts patterns from history of user actions
- Prunes patterns which are:
  - shorter than 3 commands
  - appear less than twice
  - other pruning
- Presents patterns as higher-level skills
- Decodes stored patterns to send to robot
Sequitur Algorithm [Nevill-Manning and Witten 1997]

- Extracts context-free grammar from sequence of discrete symbols
- Applied to J.S. Bach:
Algorithm 1 – Sequitur (Nevill-Manning) Algorithm

Sequitur Algorithm [Nevill-Manning and Witten 1997]

- User input:
  
  ![Input](image)

- Sequitur returns:
  
  R0 -> R1 c R2 0 R2 c R3
  R1 -> a b
  R2 -> R3 R1
  R3 -> d e f

- After pruning user would see only R2, R3
Maximum-length chaining:

- Stochastic model helps deal with input noise
- Prefers long chains for maximum throughput increase
- Intuition: iteratively concatenate to observed sequences while next input can be reliably* predicted
  - Example: if 'c' always appears after 'a b', then discard 'a b' and begin again with 'a b c'
  - *'Reliable' determined by a probability threshold
Maximum-length chaining:

- User input:
  
  a b c d e f a b
  d e f a b c d e f

- Max-chaining returns:
  
  d e f a b
  a b c d e f

- Recognizes intended control sequence
  
  a b c d e f
The Experiment

Ingredient mixing task w/simulated PR2

Example GUI screen; user sees robot's view, stimuli on perimeter of screen
Multi-phase experiment:

- Each user given two recipes to mix
- In each phase, the user mixes both recipes
- Order they are mixed varies
- Four phases per experiment
- User runs experiment twice – once with each algorithm
- Abstracted commands make later phases easier
Example Results

Recipes: Green, Blue, Yellow | Red, Blue, Yellow

Phase 1:

Phase 2:

Total 24 commands

Total 10 commands
Simulated user experiment:

- 1000 simulated users run the same set of experiments
  - 500 average noise, 500 high noise
- High noise = each command 2.5 times higher probability of mistake
Simulated Users With Noise - Actions Sent to Robot

Phase

Actions Made

0 5 10 15 20 25 30 35

1 2 3 4

High Noise – Sequitur
High Noise – Max Chaining
Typical Noise – Sequitur
Typical Noise – Max Chaining
Improvements / Challenges

• Scalability:
  • Simple demonstration showed short, simple skills
  • HCI issue: how do we present longer, more complicated skills?
  • How do we make use of the state space?

• Dealing with contingency:
  • What if something goes wrong during execution?
Fin.