EvBot Robot Software
(Machine Learned Control for Autonomous Robot Colony Research)

Presented by:

Dr. Edward Grant
Associate Professor and Director, CRIM
North Carolina State University

Department of Electrical and Computer Engineering

Presentation Outline

• Background
• Design Specifications and Implementation
• Experiments
• Conclusion and Suggestions for Future Work
Background

- **Artificial evolution** was applied to evolve neural networks to control **autonomous mobile robots**
- The robot controllers were evolved to play a competitive team game: **Capture the Flag**
- During artificial evolution, selection of the fittest was based on the results of **robot-robot competition**
- Evolved controllers were **tested in competition** against a **knowledge-based controller** and found to be able to win a small majority of games in extensive tournaments

Remote Control

- **Remote vision processing and control on a PC running MATLAB**
MATLAB Control

- Start web cam as a separate process before proceeding
- Initialize serial port for sending data wirelessly

Web Cam Process
- Grab image from JE video transmission
- Save image
- Wait one second

Main Control Loop
- Read image saved by web cam
- Process image
- Send JE command(s) wirelessly
- Wait for JE to respond
- Wait for next image from web cam

EvBot Specification

- Small size and low cost
- Ease of software development
- High-bandwidth communications
- Powerful CPU
- Advanced sensing capabilities
MATLAB for Software Development

• Very high-level language
• Extensible
• Frequently used for controller design
• Significant hardware and software requirements

Communications

• Radio frequency wireless communications
  ▪ Shared bandwidth a potential problem
  ▪ Short-range local communication
• Information sharing protocol
• Remote monitoring and control software
• Wireless Ethernet
Pentium-Class CPU

- Run complex controllers
- Run MATLAB
  - PC-compatible
  - Fast
  - Diverse sensors
  - New sensors
  - Device-driver interface

EvBot Hardware
**EvBot Software**

- BasicX controller code and interface protocol
- Custom miniature Linux distribution
- User-supplied EvBot controller

**Evolutionary Robotics (ER)**

- Roots in Evolutionary Computation, Artificial Life and Behavioral Robotics.
- Population based artificial evolution
- Automatically synthesize intelligent robot controllers
- Reinforcement Learning
- Synthesis vs. Optimization
- Behavioral vs. Dynamic systems
Population-Based Artificial Evolution in ER

Population Initialization: P(k=0)

Performance of controllers (p in P) instantiated in robots in an Environment

Fitness Evaluation of each p in P based on Performance in Environment

Re-order P based on fitness values (F(p1) > F(p2) > F(p3) > ...)

Propagate P(k) to P(k+1) using a Genetic Algorithm (GA) (mutation/crossover)

Examples of ER Work: Simple

- 1995: Jakobi, Harvey, Husbands
  - Phototaxis, obstacle avoidance (Khepera Robots)
  - Evolution in simulation with transference to reality
Examples of ER Work: Complex

- 1997: Nolfi
  - Garbage collection
  - Khepera Robot
  - Evolution in simulation with transference to reality

Examples of ER Work: Co-Competitive

- 1996: Cliff and Miller
  - Co-evolution of predator and prey
  - Khepera Robot
  - Evolution in simulation with transference to reality
CRIM ER Test-Bed: EvBots

EvBot with tactile sensors

EvBots with cameras and colored shells

EvBot II

CRIM ER Test-Bed: Environment
CRIM ER Test-Bed: Video Range Emulation Sensors

CRIM ER Test-Bed: Real vs. Simulation

Real sensors  Simulated sensors
Evolutionary ANN Controller Architecture

- Weights, connections, and topology can all be mutated during evolution

Current Issues in ER

- Control Architecture
- Evolution in simulation vs. evolution in real robots
- Generalization of ER methods to evolve complex behaviors
- Fitness selection in artificial evolution
Fitness Selection Functions

Population Initialization  
Performance of controllers $\{p_i\}$ instantiated in robots in an Environment  
Fitness Evaluation of each $p_i$ in $P$ based on Performance in Environment  
Re-order $P$ based on fitness values $F(p_i) > F(p_j) > F(p_k) > \ldots$  
Propagate $P(k)$ to $P(k+1)$ using a Genetic Algorithm (GA) (mutation/crossover)

Fitness Selection Function: $F(p)$

Hand-formulated Task-specific Fitness Functions.  
EX: Object avoidance  
$F(p) = \alpha \int (V_{left} + V_{right}) dt - \beta \int (Sensor_{act}) dt$

Co-Competitive Fitness Functions. Introduces changes in the fitness landscape (The Red Queen Effect), example Predator and Prey  
$F_{rabbit}(p) = \max \int Dist(fox, rabbit) dt$  
$F_{fox}(p) = \min \int Dist(fox, rabbit) dt$

Aggregate Fitness Functions  
$F(p) = \begin{cases} 
1 & \text{if success} \\
0 & \text{if failure} 
\end{cases}$
EvBot Capture the Flag

- Populations of robot controllers evolved to play *Capture the Flag*
- Selection based on competitive tournaments
- Games played in maze environments

Motivation

- Goal: Apply competitive fitness selection to ER
  - Controllers actively compete during evolution
  - Competition drives the evolution of complex behavior
  - Relative win/lose competition must be used to gain the benefits of competitive selection
- Problem: Initial populations are too unfit
  - Randomly initialized controllers can’t win any games at all (The Bootstrap Problem)
  - Need to use human bias to overcome the Bootstrap Problem
  - Bias will disrupt the relative win/lose competition
- Solution: The bimodal fitness function
The Bimodal Fitness Function

- Fitness $F(p)$ of an individual $p$ in an evolving population $P$ ($p \in P$) takes the general form:

$$F(p) = F_{\text{mode}_1}(p) \oplus F_{\text{mode}_2}(p)$$

- Mode 1: Accommodates sub-minimally competent initial populations (the Bootstrap Problem)
- Mode 2: Allows for competitive win/lose selection in minimally competent populations

---

The Bimodal Fitness Function

- **Mode 1:**

$$F_{\text{mode}_1} = F_{\text{dist}} + s + m$$

$$F_{\text{dist}} = \begin{cases} \alpha (D - d) & \text{if } d < D \\ 0 & \text{otherwise} \end{cases}$$
The Bimodal Fitness Function

- **Mode 2:** \( F_{\text{mode\_2}} (p) \):

<table>
<thead>
<tr>
<th>Game Pair Outcomes</th>
<th>Fitness Points Awarded</th>
</tr>
</thead>
<tbody>
<tr>
<td>win-win</td>
<td>3</td>
</tr>
<tr>
<td>win-draw</td>
<td>1</td>
</tr>
<tr>
<td>win-lose</td>
<td>.5</td>
</tr>
<tr>
<td>draw-draw</td>
<td>0 ( (F_{\text{mode_1}} \text{ dominates}) )</td>
</tr>
<tr>
<td>draw-lose</td>
<td>0 ( (F_{\text{mode_1}} \text{ dominates}) )</td>
</tr>
<tr>
<td>lose-lose</td>
<td>0 ( (F_{\text{mode_1}} \text{ dominates}) )</td>
</tr>
</tbody>
</table>

Results 1

- Game in a simple simulated world
- Robots use evolved ANN controllers
- Winner: Red
Results 2

- Game in a large complicated simulated world
- Robots use evolved controllers
- Winner: Green

Transfer to the Real World

- Game in the real world
- Robots use evolved ANN controllers
- Winner: Red
Acknowledgements

- **NC State University:**
  Dr. Edward Grant (PI), John Galeotti, Lt. Stacey Rhody (USMC), Dr. Andrew Nelson, Leonardo Mattos, Greg Barlow, Kyle Luthy, Chris Braly

- **San Diego State University:**
  Dr. Gordon Lee (PI), Damion Gastelum, Cmdr. Tom Jones (USN)

- **DARPA:**
  MTO Division, Dr. Elana Ethridge, for the original inspiration