A Case-based Reasoning Approach to Imitating RoboCup Players

Michael W. Floyd, Babak Esfandiari and Kevin Lam

FLAIRS 21
May 15, 2008
Overview

1) Problem and Motivation
2) RoboCup Simulation League
3) Related Work
4) Contributions
5) Case-based Imitation
6) Conclusions
Problem and Motivation

• Programming by demonstration:
  o imitate and perform arbitrary behaviour
  o remove need for programming expertise

• RoboCup an interesting benchmark
  o quite realistic (noise, world model, etc.)
  o data easy to generate and access
  o spatial knowledge representation
  o use imitation to create sparring partners
RoboCup Simulation League

Team 1

Client
Client
Client
Client

Soccer Server

Client
Client
Client

Team 2

Client
Client
Client
Clie
Related Work

- CBR for coordinating 4-legged team of RoboCup robots (Ros et al. 2006/2007)

- CBR for high-level soccer activities (Wendler & Lenz 1998; Marling et al. 2003; Steffens 2004; Berger et al. 2007)

- Imitation in first person shooter games (Bauckhage et al. 2004)
Contributions

• a RoboCup player that uses CBR to imitate other players

• algorithms that enable CBR to be performed by imitative agents

• an automated process for acquiring case data, learning feature weights and performing imitation
Agent Imitation

- directly imitate another agents
- goal of being "as good as", not better
- uses raw sensory data, unfiltered, to avoid adding a human bias
Automatic Case Acquisition

stores player-server communication
Case Representation

Sensory Inputs + Actions

{ Kick, Dash, TurnNeck, Turn, Catch }
Imitation Cycle

1. Message from server
   - Convert To Case
2. Retrieve Case
3. Case Base
4. Perform Action
   - Message to server
Case Base Search

• Uses a k-Nearest Neighbour search

• Calculates the distance between two cases by:

\[ d(c_1, c_2) = \sum_{n} \left[ w_n \ast d(obj_n, mat(obj_n)) \right] \]  

\[ c_1 = \{ obj_1, obj_2, \ldots, obj_n \}, \]

\[ c_2 = \{ mat(obj_1), mat(obj_2), \ldots, mat(obj_n) \} \]

mat(obj) : returns the object that “matches” with obj or nil if there is no matching object (in which case a penalty is applied)
Object Matching

- case represents "field of vision" - incomplete view of the world
- objects may be indistinguishable
Object Weighting

- automatically calculate optimum object weights
- uses a genetic algorithm
  - each gene represents weight for an object type
  - uses crossover and mutation operators
  - attempts to maximize the imitative behaviour during validation
Results

• in-game behaviour very similar when imitating simple agents

• able to automatically determine which objects are important to the imitated agent

• Object weighting occasionally found “unexpected” weights

<table>
<thead>
<tr>
<th></th>
<th>Accuracy (Calculated)</th>
<th>Accuracy (Equal)</th>
<th>Accuracy (Expert)</th>
<th>Recall kick/dash/turn (Calculated)</th>
<th>Recall kick/dash/turn (Equal)</th>
<th>Recall kick/dash/turn (Expert)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Krislet</td>
<td>72.83%</td>
<td>68.37%</td>
<td>70.27%</td>
<td>0.51/0.84/0.52</td>
<td>0.11/0.80/0.32</td>
<td>0.16/0.82/0.39</td>
</tr>
<tr>
<td>NewKrislet</td>
<td>71.13%</td>
<td>65.70%</td>
<td>69.07%</td>
<td>0.46/0.83/0.53</td>
<td>0.23/0.80/0.43</td>
<td>0.43/0.80/0.46</td>
</tr>
<tr>
<td>CMUnited</td>
<td>53.93%</td>
<td>41.90%</td>
<td>51.23%</td>
<td>0.28/0.40/0.63</td>
<td>0.06/0.23/0.41</td>
<td>0.16/0.38/0.41</td>
</tr>
</tbody>
</table>
Demo – No Learning
Demo – Imitating Single Game
Conclusions and Future Work

- successful for simple, single-state agents
- requires minimal human intervention
- future work will focus on more complex agents - multi-state and inter-agent communication
- Project Website: [http://rcscene.sf.net](http://rcscene.sf.net)