Mobile Robot Programming for Education

Jean-François Lalonde, Christopher Bartley, and Illah Nourbakhsh

The Robotics Institute
Carnegie Mellon University
Mobile Robot Programming Laboratory class
  - Course overview

Robotics concepts
  - Fundamental
  - Advanced

Educational concepts
  - Evolution over 11 years
Plan

- Mobile Robot Programming Laboratory class
  - Course overview

- Robotics concepts
  - Fundamental
  - Advanced

- Educational concepts
  - Evolution over 11 years
Course description

Mobile Robot Programming Laboratory
- 11 years at Carnegie Mellon
- Undergraduates & Graduates
- ~30 students, 2 TAs

Goal
- Students learn how to program robots!
- Weekly assignments
  - Increasingly difficult
  - Build on previous
Platform

- USB camera
- Dell Laptop
  Java 1.4.2
  Windows XP
- Wireless network adapter
- Nomad Scout
  Differential drive
  robot
- 16 sonars for localization

Students can focus on *programming*
Curriculum

1. Introduction
2. Feedback and reactive control
3. Sensor interpretation
4. Abstract action
5. Executing plans
6. Planning
7. Programming architectures
8. Single-robot game playing
9. Cooperation
10. Final competition

- PID controller
- Basic actions: WhatdoISee, GotoNextNode
- Execute a given sequence of actions
- Generate a sequence of actions
- Introduce the game
- Multiplayer game
Plan

- Mobile Robot Programming Laboratory class
  - Course overview

- Robotics concepts
  - Fundamental
  - Advanced

- Educational concepts
  - Evolution over 11 years
Robotics: Fundamental concepts

- **Perception**
  - Sonars: localization in maze
  - Camera: lighting-insensitive color detection

- **Action**
  - PI/PID controllers
  - Movement in the maze

- **Cognition**
  - Planning
    - DFS, BFS, AND-OR, etc.
Robotics: Advanced concepts

Robot observability
- Degree to which outside observer can identify the evolution of the internal state of a robot
  - Audio: speech synthesizer
  - Visual: graphical display, logging mechanism
- All teams use at least 1 form of interface
- 86% of students → very useful

Others
- Control architectures
- Reinforcement learning
- Multi-robot coordination
Plan

- Mobile Robot Programming Laboratory class
  - Course overview

- Robotics concepts
  - Fundamental
  - Advanced

- Educational concepts
  - Evolution over 11 years
## Evolution – Hardware

<table>
<thead>
<tr>
<th>Year</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>Nomadic Tech. Serial 1 &amp; 2</td>
</tr>
<tr>
<td>3-6</td>
<td>Nomad 150</td>
</tr>
<tr>
<td>7-11</td>
<td>Nomad Scout</td>
</tr>
</tbody>
</table>

### 3-wheels synch, infrared
- Wheels turn independently of body
  - Infrared sensors

### 3-wheels synch, sonars
- Independent sensor turret
  + Higher DOF
  - Large size

### Diff-drive, sonars
- Smaller size
  - Lower DOF

**Higher DOF → Higher number of possible solutions**
## Evolution – Programming environment

<table>
<thead>
<tr>
<th>Year</th>
<th>Programming environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>LISP on Macintosh</td>
</tr>
<tr>
<td>3-4</td>
<td>LISP on Windows</td>
</tr>
<tr>
<td>5-7</td>
<td>C/C++ on Windows</td>
</tr>
<tr>
<td>8-11</td>
<td>Java on Windows</td>
</tr>
</tbody>
</table>

- **LISP**
  - Command-line debugger
  - Diagnostic tool for code fragments
  - No IDE under Windows

- **C/C++**
  - Popular
  - Memory/pointers problems
  - Steep learning curve

- **Java**
  - No memory problems
  - Easy graphical interface
  - Well documented

**Need readily available, fast debugging tools**
Evolution – Final challenge

- One-on-one in shared maze
  - Interference in infrared sensors
- Two-vs-two in shared maze
- Two-vs-two in separate maze

Pick-up "gold" pieces

Must be challenging, but feasible
Evolution – Teamwork

- Great teamwork opportunity
- 3 members is the best
  - > 3: splits within teams, members left out
  - < 3: not enough to complete tasks
- Same-gender teams are more efficient
  - Students also feel that way
Conclusion

- Mobile Robot Programming Laboratory class
- Students learn
  - Fundamental & advanced robotics concepts
- Important points
  - Enough flexibility to allow creative solutions
  - Readily available debugging tools
  - Challenging but reachable problems
  - Small, well-balanced teams work best
Thank you!

Are there any questions?