

Pyro: A Tool for Teaching Robotics and AI

Douglas Blank, Bryn Mawr College
Holly Yanco, UMass Lowell

AAAI-05

This work is supported by the National Science Foundation: “Beyond
LEGOs: Hardware, Software, and Curriculum for the Next
Generation Robot Laboratory,” NSF CCLI DUE-0231363.



What is Pyro?

- **Python Robotics**
- Programming environment for advanced topics
 - Mobile Robotics
 - Artificial Intelligence
- Architecture independent
 - Robot architectures are often robot specific
 - Architectures are often difficult to learn
 - Architectures are often VERY different from each other
- Powerful research tool
- Open source
 - Easy to add functionality
 - Easy to study underlying system



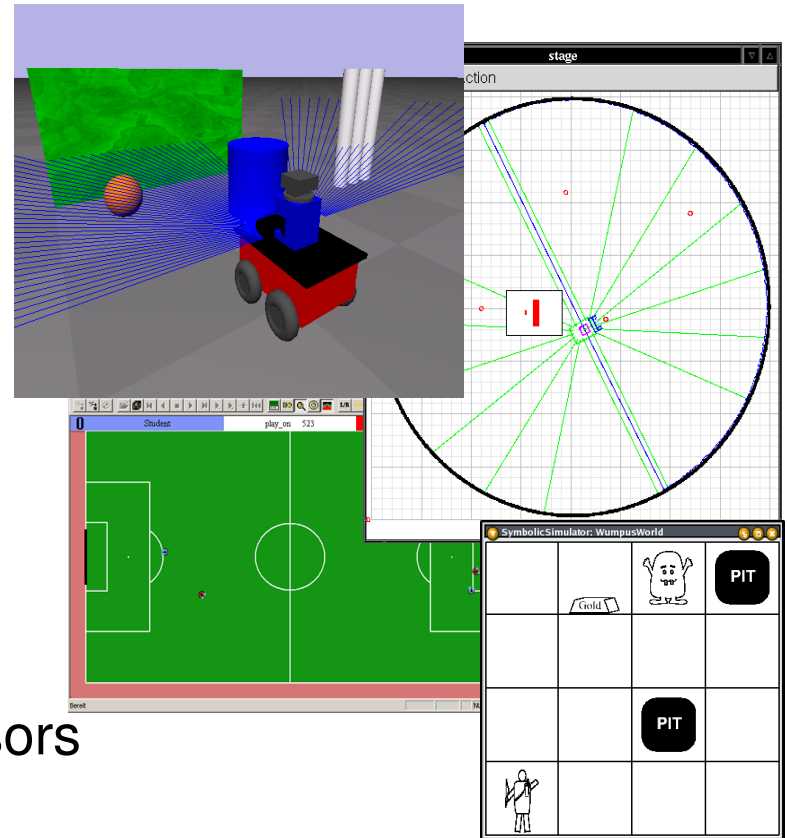
Supported Robots

- ActivMedia
 - Pioneer Robots
 - PeopleBot
- K-Team
 - Khepera
 - Hemisson
- Evolution Robotics
 - ER1
- Sony
 - Aibo Robots
- Others
 - Easy to add support for new robots



Supported Simulators

- Stage
 - Low-fidelity 2D simulator
 - Can simulate a large number of robots
- Gazebo
 - High-fidelity 3D simulator
 - Simulates Physics; displays with OpenGL
- RoboCup
 - Simulates RoboCup Soccer
- Khepera
 - Simulates Khepera with light sensors
- Symbolic
 - Discrete action/sensor simulator



Pyro Abstractions

- Default range sensor: `robot.range`
 - Can be IR, sonar, laser, etc.
- Default range units are “ROBOTS”:
`robot.range.units`
 - 1 “robot” is the length of the robot being used
- Named sensor groups: `robot.range[“left-front”]`
- Generalized motion control: `robot.move(t, r)`
- Abstract devices: `robot.gripper[0].open()`
 - Used to control devices, sensors, or visualizations

Abstractions for Portability

- `robot.range["left-front"]` might be three very different readings on different robots:
 - `robot.laser[2][4]`
 - `robot.sonar[0][3, 5, 8]`
 - `robot.ir[1][4:7]`
- and all range values could be relative

Direct Control Brain

```
from pyrobot.brain import Brain

class WallFollow(Brain):
    # follows walls on its left, ignores sonar sensors on its right
    def wallFollow(self, dist):
        frontLeft = min([s.value for s in self.robot.range["front-left"]])
        backLeft = min([s.value for s in self.robot.range["back-left"]])
        front = min([s.value for s in self.robot.range["front"]])
        if front < dist:
            print "wall in front"
            self.move(0,-0.5)
        elif (frontLeft < dist or backLeft < dist):
            print "following:",
            if frontLeft < backLeft:
                print "turn slight away"
                self.move(0.3,-0.1)
            else:
                print "turn slight toward"
                self.move(0.3,0.1)
        else:
            print "find wall"
            self.move(0.3,0)
    def step(self):
        self.wallFollow(1)

def INIT(engine):
    return WallFollow('WallFollow', engine)
```



GA + NN = Evolvable Robot

Combining the Genetic Algorithm with the Neural Network creates an easy to evolve robot controller

- 1) Evolve a list of floating point numbers
- 2) Load as weights in a neural network controller
- 3) Let robot run for a while; score performance
- 4) Performance is fitness for that “gene”

Pyro Materials

www.pyrorobotics.org

- Software downloads
- Bootable LiveCD ISO
- Curriculum
- Community



Project Support

This work is supported by the National Science Foundation: “Beyond LEGOs: Hardware, Software, and Curriculum for the Next Generation Robot Laboratory,” NSF CCLI DUE-0231363.

Principal Investigators

Douglas Blank, Bryn Mawr College

Kurt Konolige, SRI International

Deepak Kumar, Bryn Mawr College

Lisa Meeden, Swarthmore College

Holly Yanco, UMass Lowell

