CS 485/511: Robot Lab
Course Introduction & Player/Stage Setup

Christopher T. Cannon
Drexel University
March 30th, 2011
Overview

• Course Introduction
• Player/Stage/Gazebo
• Installation
• Configuration
• Writing Clients
• Demo
• Future Assignments
Course Introduction

• Both online and in-person instruction.
• All class communication should be done on BBVista.
• Design, analysis, and implementation of multi-robot systems in simulation.
• Weekly programming assignments.
Simulation

• Approximate of the real-world.
• Why would we want to do this?
  – Faster, easier and cheaper to create;
  – Easier to experiment with and demonstrate;
  – Experimental repeatability.
• What are some draw backs of simulations?
  – Not reality;
  – Harder to transition to real life (although Player makes it a bit easier).
Software Overview

• Player:
  – A server, that provides an abstraction layer to robots.

• Stage:
  – A simulator;
  – 2D;
  – Low fidelity, but can emulate more virtual robots.

• Gazebo:
  – Another simulator;
  – 3D;
  – High fidelity, but can not emulate as many virtual robots.
The Big Picture

Client -> Player

Player -> Gazebo

Gazebo -> Stage

Stage -> Real Hardware

**Slide taken from a lecture by Nate Koenig at USC**
What is Player?

• Hardware abstraction layer for robots.
• Implements a client/server model.
• Communication over TCP sockets using the Player protocol.
• Officially supported client libraries:
  – C, C++, Python
• Unofficially supported client libraries:
  – Java, LISP, Matlab
Partial List of Supported Robotics Hardware

- **Manufacturer Device(s) Driver**
  - [Acroname](https://www.acroname.com) Garcia
  - [Botrics](https://www.botrics.com) Obot d100
  - [Evolution Robotics](https://www.robosoft.com) ER1 and ERSDK robots
  - [iRobot](https://www.irobot.com) Roomba vaccuming robot
  - [K-Team](https://www.k-team.com) Robotics Extension Board ([REB](https://www.k-team.com/reb)) attached to Kameleon 376BC
  - [K-Team](https://www.k-team.com) Khephera
  - [MobileRobots](https://www.mobilerobots.com) (formerly ActivMedia) PSOS/P2OS/AROS-based robots (e.g., [Pioneer](https://www.pioneer.com), [AmigoBot](https://www.amigobot.com)) and integrated accessories, including a [CMUcam](https://cmucam.org) connected to the AUX port.
  - Nomadics NOMAD200 (and possibly related) mobile robots
  - RWI/iRobot RFLEX-based robots (e.g., B21r, ATRV Jr) and integrated accessories.
  - Segway Robotic Mobility Platform (RMP), a custom-modified version of the Human Transport (HT)
  - [UPenn GRASP](https://www.grasp.upenn.edu) Clodbuster
  - [Videre Design](https://www.videre.com) ERRATIC mobile robot platform
  - White Box Robotics 914 PC-BOT

- Also supports numerous devices (sensors, actuators), software (e.g.: for speech recognition), algorithms (e.g.: obstacle detection and avoidance), and simulators (which we talk about next).

* This list is taken from the Player Project Wiki.
What is Stage?
What is Gazebo?
Player/Stage Installation

• I recommend you install it on Ubuntu 10.10.
• Other “possibilities” are Mac OS X or Windows.
• I recommend you install both Player and Stage from source to ensure compatibility.
• If your host operating system is not Ubuntu 10.10, I suggest you install it as a virtual machine (VM).
  • Use VirtualBox an open source VM creator.
• This is going to take awhile, budget 1-to-2 hours.

04/01/10
Player Installation

1. Download Player 2.1.3 from SourceForge
2. tar xvf player-2.1.3.tar.gz
3. sudo apt-get install g++ libltdl-dev libboost-all-dev swig libjpeg-dev
4. cd player-2.1.3
5. ./configure
6. sudo make
7. sudo make install
8. export LD_LIBRARY_PATH=/usr/local/lib: $LD_LIBRARY_PATH

04/01/10
Stage Installation

1. Download Stage 2.1.1 from SourceForge
2. tar xvf stage-2.1.1.tar.gz
3. sudo apt-get install cmake libfltk-dev libpng-dev libglu1-mesa-dev libgtk2.0-dev libgtkgl2.0-1 libgtkgl2.0-dev
4. cd stage-2.1.1
5. ./configure
6. sudo make
7. sudo make install
Was the Installation Successful?

- Run “player”
  - Should see the usage and version number.
- Run “player worlds/simple.cfg”
  - A map with a robot should appear.
Defining a Robot

• Three things:
  – Interfaces: A set way for drivers to send / receive data.
  – Drivers: Piece of code that talks to hardware (laser, camera, transporter).
  – Devices: A driver, bound to a particular interface so that Player can communicate with it.
Configuration Files

• For details about any configuration issues, check the Player website, or Google “player stage tutorial” and read the first hit (a PDF by Jennifer Owen)

• There are three main types of configuration files in Player / Stage:
  – .inc: Use these files to define objects that can appear in worlds.
  – .world: This describes everything in the world (robot, objects, their layout, etc)
  – .cfg: This describes your robot, what drivers it has (which will always be stage in this class), how to interface with each item, etc.
Example: map.inc

1. define map model
2. ( 
3.   # sombre, sensible, artistic 
4.     color "black"

5.   # most maps will need a bounding box 
6.     boundary 1 

7.     gui_nose 0 
8.     gui_grid 1 
9.     gui_movemask 0 
10.    gui_outline 0 

11.    gripper_return 0 
12. )
Example: maze-laser.world

1. # defines Roomba-like robots
2. include "roomba.inc"

3. # defines Roomba-like IR sensor
4. include "sick.inc"

5. # defines 'map' object used for floorplans
6. include "map.inc"

7. # size of the world in meters
8. size [50 50]

9. # set the resolution of the underlying raytrace model in meters
10. resolution 0.02

11. # update the screen every 10ms (we need fast update for the stest demo)
12. gui_interval 20

1. # configure the GUI window
2. window
3. ( 
4.  size [ 800.000 600.000 ]
5.  center [-23.0 23.0]
6.  scale 0.02
7.  )

8. # load an environment bitmap
9. map
10. ( 
11.  bitmap "maze.png"
12.  size [50 50]
13.  name "maze"
14.  )

15. # create a robot
16. pioneer2dx
17. ( 
18.  name "robot1"
19.  color "red"
20.  pose [-23.0 23.0 0.0]
21.  sick_laser(samples 361 laser_sample_skip 4)
22.  )
Example: maze-laser.cfg

1. # load the Stage plugin simulation driver
2. driver
3. (
4.  name "stage"
5.  provides ["simulation:0"
6.  ]
7.  plugin "libstageplugin"
8. )

7. # load the named file into the simulator
8.  worldfile "maze-laser.world"
9. )

1. driver
2. ( 
3.  name "stage"
4.  provides ["map:0"]
5.  model "maze"
6. )

7. # Create a Stage driver and attach position2d and laser interfaces
8. # to the model "robot1"
9. driver
10. ( 
11.  name "stage"
12.  provides [ "position2d:0" "laser:0"]
13.  model "robot1"
14. )
Map Descriptions

• Located in the world file;
• Accepts bitmaps;
• Black pixels are considered walls, everything else is ignored;
• Also describes the size of the simulation (the unit is meters).
Client Code

• After Player/Stage is installed and running, you need to start some client code for it to do something.

• I recommend using C, C++, or Python.
  • Previous classes have used Java, but finding a version of Player and Stage that works with the Java client library may be difficult.
C++ Client Code

1. `#include <iostream>`
2. `#include <libplayerc++/playerc++.h>`
3. `int main(int argc, char *argv[])`
4. `{`
5. `using namespace PlayerCc;`
6. `PlayerClient robot("localhost");`
7. `SonarProxy sp(&robot,0);`
8. `Position2dProxy pp(&robot,0);`
C++ Client Code

1.  for(;;){
2.   double turnrate, speed;
3.   // read from the proxies
4.   robot.Read();
5.   // print out sonars for fun
6.   std::cout << sp << std::endl;
7.   // do simple collision avoidance
8.   if((sp[0] + sp[1]) < (sp[6] + sp[7]))
9.     turnrate = dtor(-20); // turn 20 degrees per second
10.   else
11.     turnrate = dtor(20);
C++ Client Code

1. if(sp[3] < 0.500)
2. speed = 0;
3. else
4. speed = 0.100;
5. // command the motors
6. pp.SetSpeed(speed, turnrate);
7. } //ends for loop
Compiling/Running

Compile:

```bash
  g++ -o simple `pkg-config --cflags playerc++` simple.cc `pkg-config --libs playerc++`
```

Run Player:

```
  Player simple.cfg
```

Run program:

```
  ./simple
```
Demo
Previous Assignment: Wall Following
Left Wall Following
Right Wall Following
Previous Assignment: Path Planning
Path Planning with Minkowski Sum
Point Robot Application
“Sofa” Robot Application
“Sofa” Robot with Extra Degree of Freedom

04/01/10
Previous Assignment:
Ant Colony Optimization

04/01/10
Initial Pheromone Trail
Optimized Pheromone Trail
Thank you for your time and attention.