## 1 Dates

Lab date: September 15, 2005

## 2 Introduction

This lab is structured to acquaint you with the subtleties involved in using the Player/Stage/Gazebo (PSG) robot interface and simulation system. Player is a network server for robot control. Player runs on board a single robot and provides a clean interface to the robot's sensors and actuators over an IP network. Stage is a 2D robot simulator suited for large populations of robots in an environment specified by a bitmap image. Gazebo is a 3D physics-based robot simulator suitable for smaller numbers of robots simulated at high fidelity. The physics for Gazebo is provided by the Open Dynamics Engine (ODE), which integrates physical dynamics for arbitrary kinematic structures through optimization.

PSG provides an infrastructure for developing robot controllers. You will write Controllers are as client programs that send control commands to and request information from a robot through its Player server. Stage and Gazebo can simulate various types of robot platforms (i.e., hardware) and populations. The same interface, provided by the Player robot server, is used to control a robot in the real world or its equivalent in a Stage/Gazebo simulation. Robot platforms that are not currently supported in PSG can developed through implementing appropriate Player server interfaces and devices in Stage or Gazebo.

Because of its flexibility and portability, PSG will give you good experience with developing robot controllers. Additionally, many interesting advanced track final projects can be implemented using PSG.

## 3 PSG Framework

As illustrated in Figure 1, PSG is a framework for robot control consisting of devices, robot servers, and robot clients

Devices (e.g., a laser, a camera, or a complete robot) are actual hardware in the real world or simulated hardware that exists in a virtual environment maintained by Stage or Gazebo.

A robot server (e.g., Player) is the information interface between the robot and any program that requests information from or sends commands to the robot. Regardless of whether a device is real or simulated, the robot server provides the same interface to the robot for client programs. Thus, controllers developed on a simulated device will immediately run the equivalent real robot device[1], given PSG support for the device.

[1]Be careful to note that controller that can run a robot does does not imply that the controller will work properly or yield expected control results

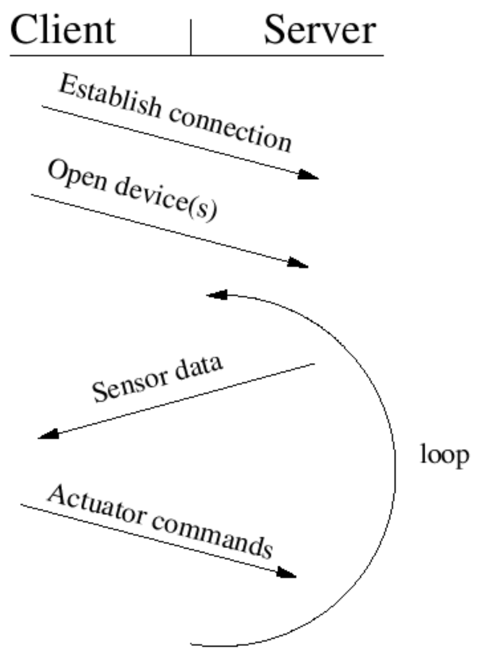

*Figure 1.1: Example client/server interaction*

Figure 1: Example PSG client/server interaction

A robot client is a user-developed program that accesses robot functions through the robot server. For controlling a single, the robot client will first establish a connection to the robot's server and then command the robot by reading data from the server and sending appropriate control command back. The job of the developer is to write control programs that produce commands that yield desired behavior from information received from the robot server[2]

Lastly, one of the major advantages of Player as a robot server is its independence from a particulare client-development language. The interaction between Player and a client program is done completely over a TCP/IP (or UDP/IP) network connection. Thus, any language with libraries that supports Player functionalities can be used to develop robot clients. The most supported client language is C (supported through libplayerc). Many other languages are supported including C++, Python, Java, and GNU Octave.

[2]Referring to the lecture on control theory, the description of the robot client should remind you of feedback control (i.e., commands [u] that produce desired state [x] given observations [y]).

## 4 Instructions

For this lab, you will be expected to work with Player/Stage/Gazebo to set up and run a very simple demonstration of PSG.

The first thing that you must do is make sure you can use the PSG tools, which are located in the /contrib/projects/psg/local/bin directory. Your PATH variable must point there for the tools to work, so add this line to your .environment file:

pathappend PATH /course/projects/psg/local/bin

You can see that this worked if you type wxgazebo into a terminal.

- Start by copying the directory /course/cs148/asgn/atrack/lab1 somewhere you can use it. In here you will see two directories, gazebo and player.
  - go into the gazebo folder and type 'wxgazebo lab1.world' You should see a world pop up with some objects and a robot in it.
  - In a new terminal, go into the player directory and type 'player -g 0 lab1.cfg' This will set up the player client on port 6665 (which is the default port).
  - In a third terminal, type 'playerv' and a window will pop up with a grid. Select the following:
    - devices–position–subscribe
    - devices–position–command
    - devices–laser–subscribe
  - Now you can see the laser data from the robot, and you can control the robot by dragging the '+' in the center around different directions. Playerv is simply a client with a gui.
- Now it is time to write our own client
  - In the directory player/client, there is a sample client. Type 'make' to compile it.
  - To run it, you must have gazebo running as well as player ('player -g 0 lab1.cfg'), and type './lab1 1 6665'
  - The '1' is a speed multiplier for the robot, and the '6665' is the port number to connect to.
  - Look through the code for this client. Right now, all this does is move the robot forward once and then quit.
- Your task for this lab is:
  - Make the robot move around this world avoiding obstacles using laser ranging data, and only quit if the robot makes it out of the arena or a command-line specified time limit is reached. Also, record data of where the robot has been to a file.

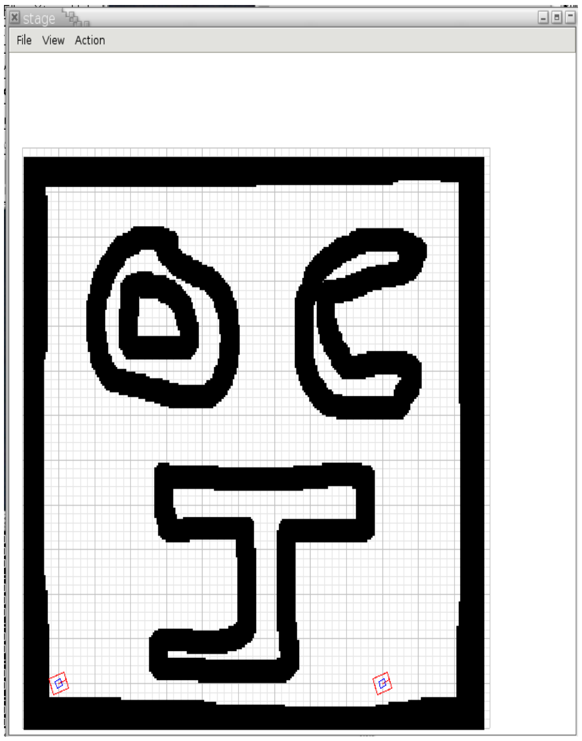

Figure 2: Stage environment (artistically created by cjenkins) with 2 robots

  - You will also have to modify the lab1.cfg file to include truth data.
  - To do these things, you may use TRUTH data, but not to avoid obstacles.
  - Documentation for this is found on the player/stage website, http://playerstage.sourceforge.net/doc/

## 5 A Few More Things

You were given this world to play around in, but what if you wanted to make your own? Look in the /lab1/worlds directory, and you will find the .jpg file that was used to make the world that you just explored. On the website: http://playerstage.sourceforge.net/doc/Gazebo-manual-0.5-html/gzbuilder.html you will find information on how to use the gzbuilder utility to make your own terrain files, which are incorporated into .world files.

- Make your own world (you can use lab1.world as a model)
  - Make a new terrain file from a new .jpg

  - Add at least TWO robots to this environment
  - write a .cfg file to go along with this world, using lab1.cfg as a model.
- Using playerv, control both robots.
- Additionally, try using your world image as a map in Stage.