1 Introduction

In the previous lab, you performed a local installation of Player/Stage/Gazebo (PSG) from which sensor information can be viewed from mouse-controlled robots. You will now use this installation to begin writing robot controllers in the client/server framework of PSG. Specifically, you will write controllers for a robot or a team of robots to explore and cover an environment.

2 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.

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.

3 Assignment

For this lab, you are expected to create new Stage and Gazebo worlds and develop client programs to drive Pioneer robots to explore these worlds. **This lab, as well as the other advanced track labs, will be due on 10/14/2004.**

[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

[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]).

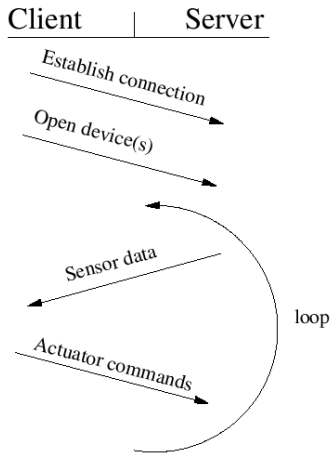

Figure 1.1: Example client/server interaction

Figure 1: Example PSG client/server interaction

However, we highly recommend completion of this lab before the assignment of the next lab (1 week from now).

This assignment involves the following tasks:

- development of robot client for Player that explores an environment
  - write a control program in the language of your choice to completely explore a given environment using one of the Pioneer robot devices
  - given command line input, the robot client should be able to use laser and/or sonar sensing
  - the robot must avoid collisions with the environment, objects, and other robots
  - the robot client must record the locations visited in the environment in a form that can be displayed after program execution. (extra respect: record the 2D polygon of areas viewed by the robot)
  - the robot client must provide a command line parameter to specify time allowed for execution in milliseconds. When this parameter is specified, the robot client must terminate after the specified time has elapsed.
  - robot control computation must use only robot sensor data. More specifically, global and absolute truth information, such as position-¿px, **cannot be used for control**. However, absolute truth information can be used for recording the performance of the robot.
- creation of three new Stage and Gazebo worlds

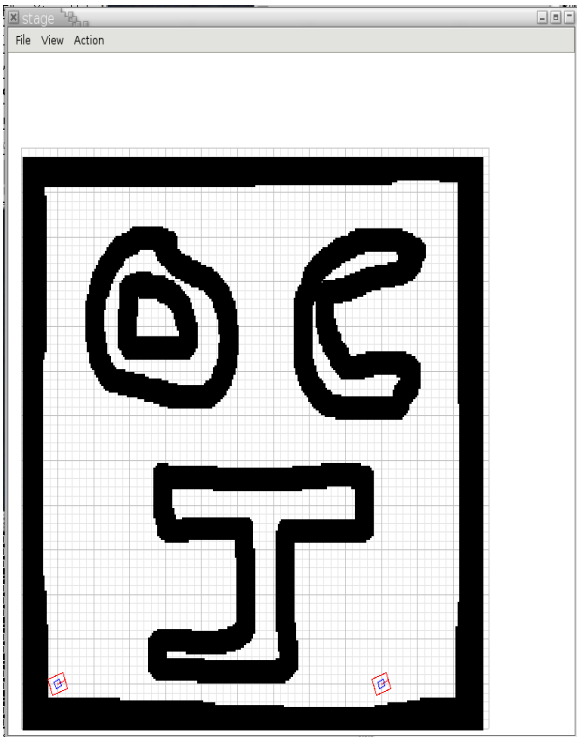

Figure 2: Stage environment created by Chad (demonstrating his lack of artistic ability) with 2 robots

  - one simple Stage world that consists of your initials (something like Figure 2) and 1 Pioneer robot
  - one sophisticated Stage world with 4 Pioneer robots to exercise the limit of capability for your exploration client
  - one Gazebo world of your own creation

- utilization of your exploration client to drive robots in the environements you have created.

Once completing this implementation, you will turnin the following:

1. the **commented** source code for your client program
   - usage for the client must be obtainable from the command line and the source file header
   - build instructions should be available from the source file header
   - one source file is preferred, but not required

2. directions, scripts, or source for displaying coverage after client execution
3. world files, images, and other necessary files for your Stage and Gazebo worlds
4. a 2-4 page report[3] describing your exploration client algorithm/implementation and how you tested your client using your sophisticated Stage world
    - one suggestion: a plot showing area covered by a robot over varying execution durations and initial starting locations

3.1 Directions for C Development

1. read over and copy the sample Player client listed at [2].
2. compile the sample client; the following worked for me:

```
gcc -I/home/cjenkins/cs148_my/local/include simple.c
  -lm /home/cjenkins/cs148_my/local/lib/libplayer* -o simple
```

3. modify the client as appropriate

4 References

1 PSG User Documentation, http://playerstage.sourceforge.net/doc/doc.html

2 Libplayerc Client Library Reference, http://playerstage.sourceforge.net/doc/Player-cvs-html/libplayerc/

[3] images not included in page count