CS 493/790(X) – Advanced Topics in Robotics

Instructor: Monica Nicolescu

Lab 1 – Handout

1. Introduction

The purpose of this lab is to get you familiar with the robot programming tools that will be used throughout the semester. In this class we will use the Player/Stage simulator, which is freely available from `http://playerstage.sourceforge.net/`.

Player is a network server for robot control. Running on your robot, Player provides an interface to the robot's sensors and actuators over the IP network. Your client program talks to Player over a TCP socket, reading data from sensors, writing commands to actuators, and configuring devices on the fly. Player supports a variety of robot hardware, in particular the ActivMedia Pioneer mobile robots.

Stage is a 2D simulator that allows for simulating large number of mobile robots, sensors and objects in a 2D bitmapped environment. Stage is most commonly used a Player plugin module, providing populations of virtual devices for Player. Users write robot controllers and sensor algorithms as 'clients' to the Player 'server'. 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 simulation, which makes it very easy to transfer the code from simulation to the real robot.

During this semester you will use Player/Stage to develop your own robot controllers. The software is already installed on the lab computers and can be also easily installed on personal machines.

2. Player/Stage Architecture

As illustrated in Figure 1, the Player/Stage framework for robot control consists of the following main components: **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.
The **server** (Player) provides the interface between the robot and any program that requests information from or sends commands to the robot. Typically, Player is executing locally on the computer to which the devices of interest are connected. In many cases, this computer is the robot itself, but it could also be, for example, a desktop machine attached to a SICK laser range-finder.

The **client** is your controller program, which produces output commands for the robot. The client can execute anywhere that has network connectivity to the machine hosting the server.

In the beginning, the client establishes a TCP socket connection to the server. The client next sends some messages to the server to *open* the devices in which the client is interested. After that, the server continuously feeds data from those devices to the client, and the client exerts control by

sending appropriate commands back to the server. It will be your job to write a controller (client) that produces the desired commands for the robot.

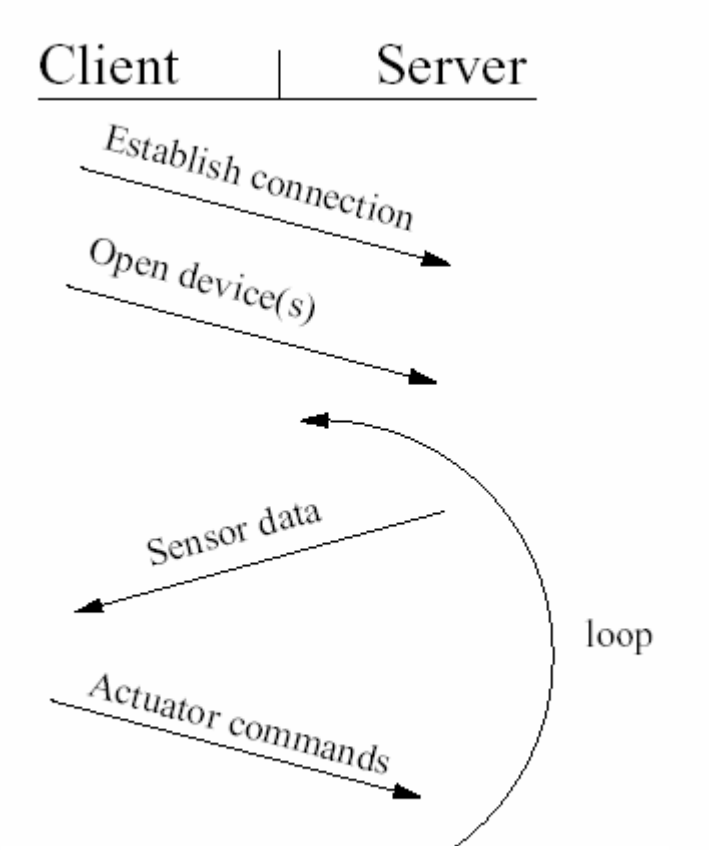

Figure 1. Player/Stage client-server interaction.

3. Getting Started

During the first lab we will experiment with basic functionality in Player/Stage, learn how to use simple tools and how to write simple controllers.

- Set the path to the Stage world directories. Add the following line to your .bashrc file:

```
> export STAGEPATH=/usr/local/PlayerStage/stage-1.6.2/worlds
```

- `Copy all the .inc files from /usr/local/PlayerStage/stage-1.6.2/worlds` to your work directory

- Download the files from `http://www.cse.unr.edu/~monica/Courses/CS493-790/Labs/` into your working directory.

- From /usr/local/PlayerStage/player-1.6.5/examples/c++/ copy to your work directory the following files: laserobstacleavoid.cc, visualservo.cc, sonarobstacleavoid.cc, randomwalk.cc. These files implement the following controllers:

- laserobstacleavoid.cc: simple obstacle avoidance using the laser
- visualservo.cc: simple color tracking
- sonarobstacleavoid.cc: simple obstacle avoidance using the sonar
- randomwalk.cc: simple wandering behavior

- Compile the examples using the provided Makefile

- Start the player server:

```
> player simple.cfg
```

A window will open with a cave-like environment and two of pioneer robots.

- You can observe the information from the robot sensors using the playerv utility. In a different terminal type

```
> playerv
```

Another window will open. From the menu experiment with subscribing to the following devices: laser, position, blobfinder, and sonar.

- With the player sever running, test the provided sample programs. Look through the code for these clients and try to understand how the information from the sensor is provided and how the commands are sent to the actuators. In particular, look at the following proxies: SonarProxy, PositionProxY, LaserProxy, BlobfinderProxy.

  You can find more details about the Player proxies at:

  http://playerstage.sourceforge.net/doc/Player-cppclient-1.5-html/node1.html

For this lab, you should make small changes to the provided controllers and observe the change in the robot's behavior. For example, you can modify the "laserobstacleavoid" client, such that your robot does not get stuck in a corner. A simple strategy to solve this problem is to use a counter that keeps track of successive left-right turns made by the robot.

- Create a new world file. In PlayerStage/stage-1.6.2/worlds/bitmaps you will find sample bitmaps for the environments used in the worlds directory. Use gimp to create a new environment of your own and incorporate that into a new .world and .cfg file. In this environment, add three robots: one yellow, one red and one green. Create a static "yellow" colored target (taking example from everything.world). You can also add more robots and static objects of various colors. Test the sample programs, and their modified versions in this new environment.

There is nothing to submit for this lab. However, for the next lab you should have available the modified controllers and the new environment.