Technische Universität München

Faculty of Informatics

Robotics and Embedded Systems

http://www6.in.tum.de May 26, 2010
Lab Course: Human Robot Interaction **Sheet 5**
Manuel Giuliani, Thomas Müller, Prof. Alois Knoll giuliani@in.tum.de, muelleth@in.tum.de

Submission Deadline: June 09, 2010

Exercise 7

Until now, we only used modules that had direct Ice connections. However, in JAST we are using the publish/subscribe service by Ice, which is called **IceStorm**. With this, all JAST input modules (object recognition, speech recognition, ...) send messages to defined topics via a topic manager and other modules can subscribe for these topics at the topic manager to retrieve the messages by the input modules. Chapter 44 of the Ice user manual gives an introduction to IceStorm.

In order that you can integrate your code into the JAST system at the end of the course, we are giving you a runtime environment, which you have to setup on your computer. The runtime environment already contains a dummy robot server and a simulated speech recognition.

Runtime Environment Installation

To install the runtime environment, please follow these steps (copied from README.txt that's also part of the runtime environment package):

```
1. Download the runtime environment from
http://www6.in.tum.de/pub/Main/TeachingSs2010LabCourseHRI/hri-lab-course-runtime.zip

2. Make sure the following tools are installed:
  - Ice 3.2.1 (http://www.zeroc.com/), we can only give support for this older
    version of Ice, since JAST runs with this version. We made the experience that
    you can install different Ice versions in parallel without running into
    problems.
  - Java 1.5 or greater (http://java.sun.com/)

3. Set environment variables:
  - Make sure the Ice utilities and Java are in the system path
  - Set a variable ICE_HOME to point to the root of the Ice
    installation (i.e., $ICE_HOME/bin/slice2java should exist)
  - On Unix, make sure that LD_LIBRARY_PATH includes the Ice libraries if they're
```

```
  not in /usr/lib (i.e., include $ICE_HOME/lib in the path)


4. Create a directory for everything to be installed in (e.g., ${HOME}/jast).
   In what follows, we'll assume that you used ${HOME}/jast; adapt
   appropriately if you put it somewhere else.

5. Extract the runtime files from the zip file to ${HOME}/jast

6. Set the JAST_RUNTIME environment variable to ${HOME}/jast/runtime (or wherever
you put the runtime).

7. Copy the JAST Ice interfaces from the folder "slice" in the zip file in
whatever directory you want.

8. Set the JAST_ICE_DIR variable to wherever you copied the ice interfaces to.


You're now ready to run things! To start the runtime up, you need to do the
following:

1. Make sure that the JAST_RUNTIME (and ICE_HOME) variables are set properly
   as described above.

2. Start the IceGrid registry:
   - cd ${JAST_RUNTIME}
   - icegridregistry --Ice.Config=config/config.grid

3. Start the IceGrid node (in another window; leave the registry alone):
   - cd ${JAST_RUNTIME}
   - icegridnode --Ice.Config=config/config.localhost

4. Start the IceGrid GUI (in yet another window):
   - cd ${JAST_RUNTIME}
   - java -jar ${ICE_HOME}/bin/IceGridGUI.jar --Ice.Config=config/config.grid

5. If you use Windows, you can also use the batch scripts we packaged with the
runtime environment.

6. Set up the IceGrid runtime to run servers:
   - Start up the registry, node, and GUI as described above
   - In the IceGrid GUI, choose "File - Login"
   - In the IceGrid GUI, choose "File - Open application from file"
   - Browse to the ${JAST_RUNTIME}/config directory and select
     "hri-labcourse-config.xml"
   - Choose "File - Save to registry"
```

```
7. Start the servers in the GUI (for example speech recognition):
  - In the GUI, choose the "Live Deployment" tab and open the "localhost"
    node
  - Right click on "SpeechRecognizer" and choose "start"
  - In the window that pops up you can now type sentences which are then sent
  over IceStorm
  - Speech recognition will print out an error message when you start it,
    this is because it cannot find the actual speech recognition program
```

Installing New Servers

When you finished installing the runtime environment, you can start to integrate your own code into it. In the following we will explain how to integrate your object inventory into the runtime, similarily you can also integrate other servers. Here are the steps to integrate your object inventory into the runtime environment:

1. The zip file with the runtime environment also contains a folder `code`, which holds an ant build file and a directory with source code. Unzip the `code` folder to any folder on your computer.
2. In the unziped source code folder there are two source files `ObjectInventoryI.java` and `ObjectInventoryApp.java`. `ObjectInventoryApp.java` does not need to be changed, it holds the code to make a connection to the JAST runtime. You have to copy your object inventory source code to `ObjectInventoryI.java`.
3. Now run the ant build file, it should create a java container `HRILabCourse.jar` and copy it to the folder `share` of the JAST runtime. To do that cd to the folder with the build file and type `ant install`. You can also use the build file to create the Ice files by typing `ant slice2java` or clean the generated files with `ant slice2java`.
4. When you compiled `HRILabCourse.jar` and copied it to the runtime environment
   - go to the IceGrid Admin window
   - click on File - Login
   - click Okay
   - click File - Open - Application from Registry and open the JAST application
   - in the appearing JAST configuration tab right-click on Nodes - localhost and add New server from Template
   - For the new server chose template JastServerJava
   - Name: ObjectInventory
   - mainClass: sheet05.ObjectInventoryApp
   - classpath: share/HRILabCourse.jar
   - endpoints: use default
   - type: ::jast::representation::ObjectInventory

- click File - Save to Registry
- click the Apply button
- change to Live Deployment tab
- there should be a new server ObjectInventory now, that can be started like the other already existing servers.

Exercise 8

Now that we have the runtime environment with the speech recognition simulator running, we will use it to command the robot to make movements and to answer questions about the objects it sees on the table. For that, you first have to write a new class that implements the interface `jast/listener/SpeechRecognitionListener.ice`. To integrate this class into the runtime environment you can copy and adjust the files we gave you to integrate the object inventory, basically you just have to replace any mention of object recognition with speech recognition. For example the new class has to listen to `::jast::listener::SpeechRecognitionListener` instead of `::jast::listener::ObjectRecognitionListener`. You will also need to write your own slice file for your class, you can place it into a new folder `jast/reasoning` since this will be your first cognitive class.

When you have the class that implements `SpeechRecognitionListener` you get all inputs from speech recognition over the method `void recognizedTurn (string top, ::Ice::StringSeq alternatives, ::jast::common::Timestamp startTime, ::jast::common::Timestamp endTime`. The paramter `top` holds the speech recognition result with the highest confidence value, `alternatives` gives you recognition results with lower confidence values.

Take the sentence from the string `top` and translate it into robot movements and queries to the object inventory. The robot should be able to understand sentences of the following form:

`verb adjective noun`

where `verb` can be:

- give me a
- show me a
- take a
- open the
- close the

`adjective` can be:

- blue
- green
- red
- yellow
- orange

- small
- medium
- big
- left
- right

and `noun` can be

- bolt
- cube
- nut
- slat
- hand

With this "grammar" you can parse sentences like "give me a green cube" or "open the left hand". When you translate these simple oders into robot movements, the robot should reason if the orders you give make any sense, for example if it gets a command "give me a yellow bolt", the robot should dispay a message "there is no reachable yellow bolt on the table" if it cannot reach any yellow bolts. Also, in the current form of our "command grammar" one can build sentences like "close the small cube" which do not make sense in the JAST context, in that case the robot should also display a message.

Notice

Please send your solution files by June 09, 2010 to `giuliani@in.tum.de` and `muelleth@in.tum.de` with subject `HRI Lab Course` as usual.