Technische Universität München

Faculty of Informatics

Robotics and Embedded Systems

http://www6.in.tum.de April 26, 2010

Lab Course: Human Robot Interaction **Sheet 1**

Manuel Giuliani, Thomas Müller, Prof. Alois Knoll giuliani@in.tum.de, muelleth@in.tum.de

Submission Deadline: 28 April 2010

The JAST Robot

Robots that are able to interact with humans usually consist of many parts: they have to be able to see and listen to their environment, they have to understand what they are sensing, they have to reason about the sensor input, and they have to figure out how to react appropriately. The robot we are using in this lab course (Figure 1) was developed for the European project JAST, which stands for Joint-Action Science and Technology. The goal of this project was to research how humans manage to work together on a common goal and how these research results can be transferred to robots. The robot has a pair of manipulator arms with grippers, mounted in a position to resemble human arms, and an animatronic talking head capable of producing facial expressions, rigid head motion, and lip-synchronised synthesised speech.

In the scenario that was developed for JAST, a human and the robot build objects from a wooden toy construction set together. Several versions of the robot were already developed and tested, for example in one version the robot explained a building plan to the human who had to build the objects by herself, in another version the robot was able to explain errors in the building plan and handed over correct assembly pieces to the human.

The software of the JAST robot is distributed to six computers that host software modules, which have diverse functions. Figure 2 shows these software modules and how they are connected.

The robot has input modules for *speech*, *object*, and *gesture recognition* as well as a *face tracking* software. The processed information of these components is collected centrally in the so-called *broker*. The interpretation module can register for data from the information modules at the broker and get notified if newly processed data arrives. Part of the information modules are *multimodal fusion*, which combines the information from speech and gesture recognition, the *world model*, which keeps track about the shape and position of all objects on the table, and the *goal inference*, which takes the information from object and speech recognition and predicts what actions the user is going to do next. The reasoning modules process the interpreted information to decide what the robot should say and do next. For that, a *dialogue manager* controls the conversation between human and robot and a *planner* keeps track of the current building plan and which steps of this plan have already been executed. Finally, the robot has components to output speech to the human and to control the body and the head. Additionally, there are some very specialised modules to produce robot output, for example the *reference generation*, which is used to generate expressions that can be used by the robot to talk about the objects in its environment.

Figure 1: The JAST robot

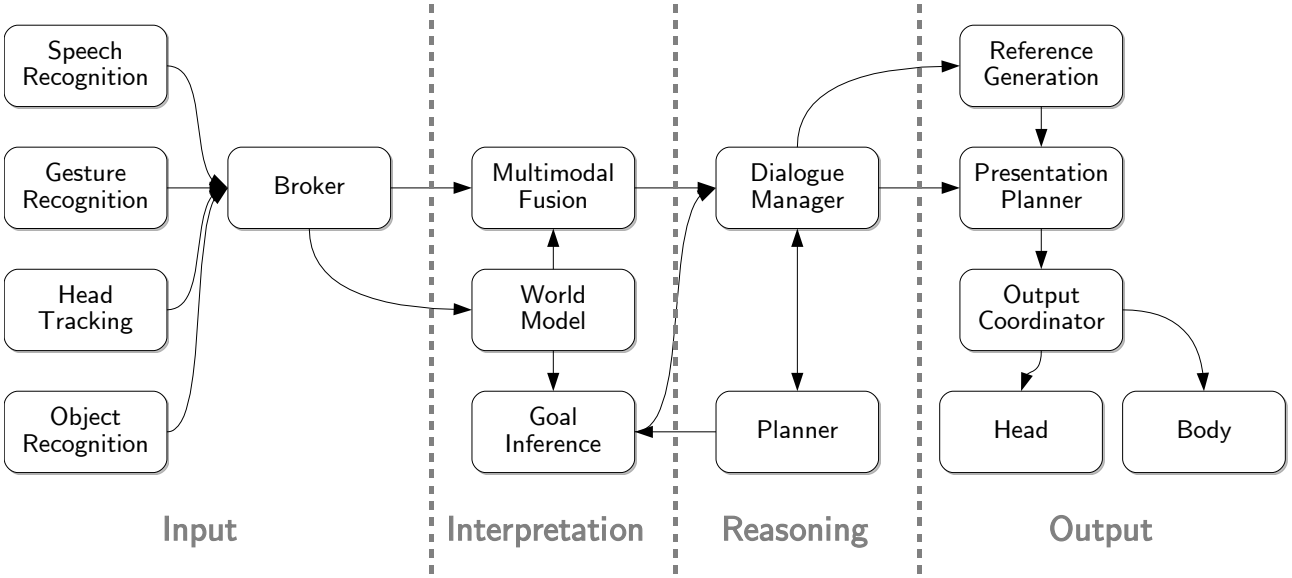

Figure 2: System architecture of the JAST robot

In this lab course, we will get to know the individual parts of the robot, we will learn how these parts are connected to each other, and most importantly we will find out how they have to be synchronised so that the robot is able to interact with a human in a reasonable way.

Exercise 1

The software components of the JAST robot communicate via a middleware called Internet Communications Engine (Ice). Ice is similar to Corba, a middleware to send messages from one computer to the other. It supports many operating systems and object-oriented programming languages. This way, developers can use their preferred OS and programming language since Ice hides the communication processes from the user. For example, in JAST we are using software modules programmed in Java, C++ and Python, running on Windows and Linux systems.

In the first exercise you will learn how to use Ice. For that, please follow the following steps:

1. Download the Ice documentation at the following URL: `http://www.zeroc.com/download/Ice/3.4/Ice-3.4.0.pdf` (if the link does not work, `http://www.zeroc.com` is the website of the company that develops Ice.)

2. We recommend to read chapters 1 and 2 of the documentation, they explain in more detail how Ice works and the main concepts behind it. If you already have experience in distributed programming, you may skip this section.

3. Install Ice on a computer of your choice. You can download everything you need for Ice at `http://www.zeroc.com` in the download section. If you do not have a suitable computer we can give you an account to one of our lab computers.

4. Chapter 3 of the Ice documentation shows an example for a client and a server that communicate via Ice. The example is presented for many programming languages; implement the example in the programming language of your choice. However, consider that the next exercises have to be programmed in Java.

5. Send your solution files (SLICE-definitions, Sourcecodes for client and server) to `giulian@in.tum.de` and `muelleth@in.tum.de` with a subject `HRI Lab Course`.