The iCub Cognitive Robotic Experiments

by
Semih Onay
Supervised by
Assistant Professor Elena Battini Sönmez

Undergraduate Program in Computer Science
Istanbul Bilgi University
2015

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

YARP	Yet Another Robot Platform
ACE	The ADAPTIVE Communication Environment
ODE	Open Dynamics Library
OpenGL	Open Graphics Library
SDL	Simple DirectMedia Layer

1. ABSTRACT

Intelligent agents with human level of proficiency is the ultimate challenge for science and technology. Especially for the artificial intelligence field. Desing of such systems now eager to evolve and behave like humans. Recent discoveries in this field are focused on training communication skills with perception, classification and action which are modelled from humans. They are systems that are capable of recieving environmental activities(stimulus) and communication. This paper presents a cognitive humanoid robot that combines abilities such as object detection, image aqusition, language and movement. Aim is to understand how motor control abilities, visio-motor abilities and linguistics are designed and developed. Study includes robotic simulation applications. Cognitive modules will be developed by using application templates and documentation.
Keywords : cognitive science, humanoid robots,iCub

2. INTRODUCTION

The iCub is a humanoid robot that developed at Istituto Italiano di Tecnologia (IIT) as part of European project RobotCub and later adopted by more than 20 laboratories arround the world. It has 53 motors that can move the head, arms and hands, midriff, and legs.It can see and hear, it has the sense of proprioception[1] . It's designed to aid studies of human cognition and artificial intelligence. Project members developed computer simulator to experiment new techniques.

Computer simulations are getting important in area robotics area. Simulations may not provide real complexity of the physical world and not reliable as real dynamics. The simulator of iCub is an easy way to test new algorithms and methods instead of dealing with complex configuration of iCub hardware. The simulator is designed to be accurate as real world psychics and dynamics. Development is based on directly from first prototype of simulation environment Webots.[2] It was expensive and had limited access to source code which made hard to modify source code in order to add some functionalties.Then iCub simulation created. Simulation environment uses to simulate body movement and collision detection algorithms to measure psychical interaction with the world. ODE[3] is used in wide range of projects like GAZEBO[4] . ODE is an open source physics engine forauthoring tools, computer games,etc.It uses OpenGL[5] renderer and it has some disadvantages due to limitation of OpenGL engine computation efficiency on complex structures. iCub simulation is using OpenGL directly via SDL[6] which helps to render complex robot movements and computationally efficient simulation observations. The aim of this study is to experiment and understand how cognitive architecture of robotics designed and implemented.

---

[1]Body configuration and movement using accelerometers and gyroscopes
[2]Webots : https://www.cyberbotics.com
[3]Open Dynamics Engine : http://www.ode.org
[4]GAZEBO : http://gazebosim.org
[5]Open Graphics Library : https://www.opengl.org
[6]Simple DirectMedia Layer : https://www.libsdl.org

3. LITERATURE REVIEW

Development of simulator is described by abstraction of parts to handle complex instructions more precisely and efficient. Some other external software libraries are used to reduce required time to animate given parts of robots body from parameters. Abstractions made it easy to implement new methods,algorithms into a simulation environment. Understanding of these libraries are the key of creating new interfaces and methods to robot in virtual world. Action primitives are pre-defined inside a simulation environment to extendcapability of creating new interfaces to virtual simulation world and can be changed or taught as different languages which helps to extend knowledge about languages.

The great challenges in robotics is to imitate human behaviour. Most of the humanoid robots have some perception to recognize their surroundings autonomosly. Grasping an object in its environment, is one of the most difficult tasks for all kinds of robots. Many parameters must be provided for, the objects, surroundings, and the specification of the assignment. Taking these parameters into consideration, the ability to receive sensing information from the robot is crucial when implementing an efficient robotic grasp. The quality of the sensing information must also be taken into consideration, as signals may limit precision and can potentially be noisy. In recent years, there have been several models implemented to perform a grasping behavior.[**vernon**]

- Knowledge based grasp action,
- Geometric grasp action,
- Sensor driven and learning based grasping.

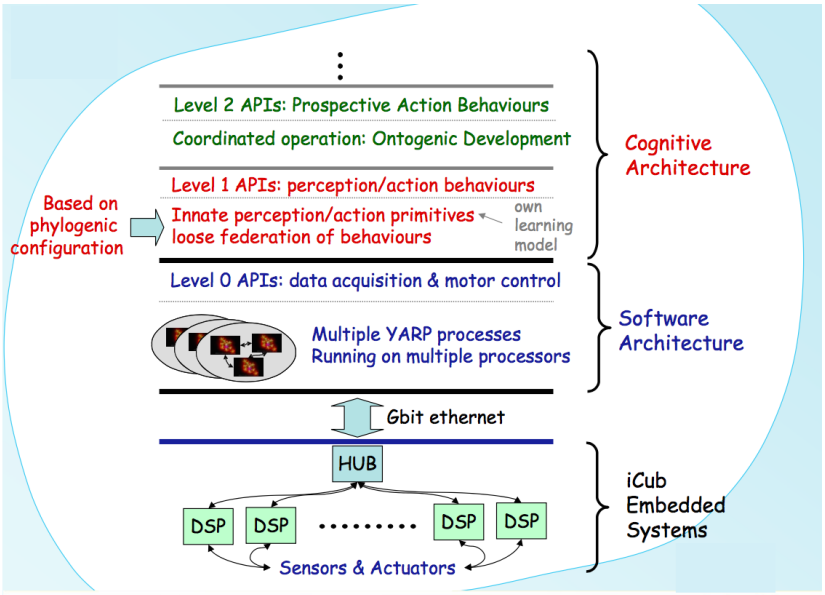

Figure 3.1. General View of iCub

4. METHODOLOGY

4.1. The iCub Cognitive Architecture

The architecture consists of modules and sub-modules. Their relations are similar to the human mental system. Gaze controling, reaching, and locomotion establishes the simple goal oriented actions. Episodic and procedural memories are effects a simple version of an internal simulation to provide capabilities for prediction and reconstruction, as well as productive model construction by experienced affordances from environmental sensors. A simple process of homeostatic state is achived by the affective state that provides action selection and attention selection. These futures are already implemented in the simulation environment. Indiviual components of the ar-

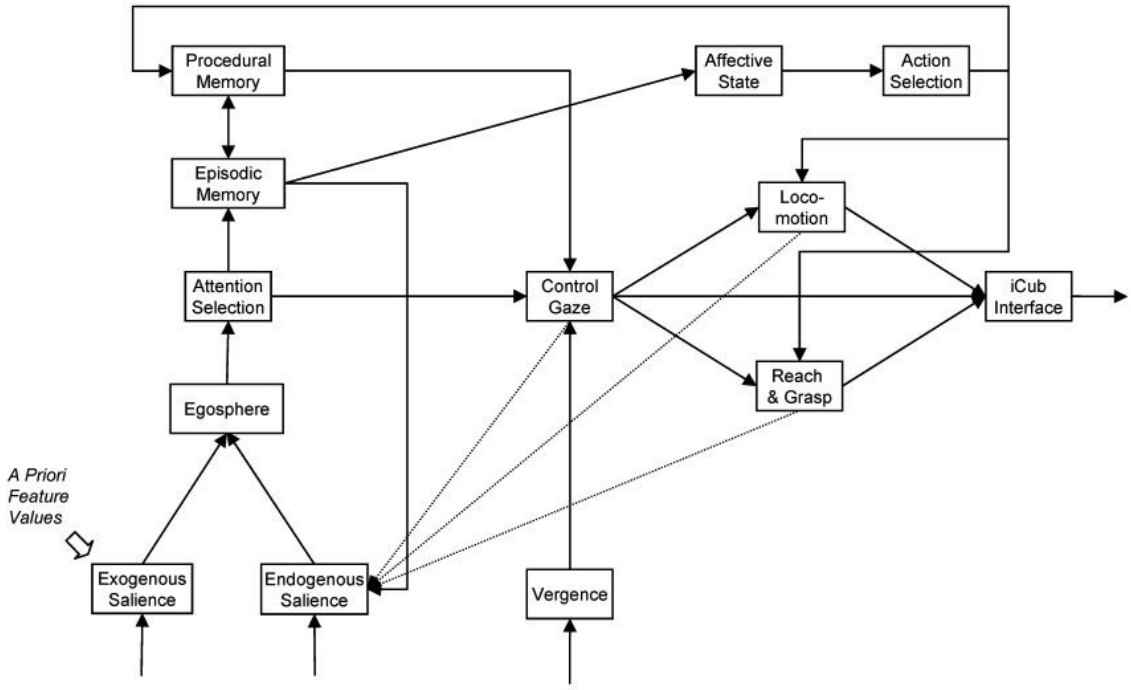

Figure 4.1. General View of iCub Cognitive Architecture

chitecture operates synchronously for a series of states representing cognitive behaviour that emerges from the interaction of separate parallel processes instead of being governed by some state machine as it like the most of cognitive architectures. In addition, motivations are encapsulated in the affective state of the system. They are designed explicitly to address curiosity and experimentation by using explorative motives which are triggered by exogenous and endogenous factors, respectively. Distinction between the exogenous and endogenous salience is represented by the need to include an attention system to consolidate both factors.

4.2. The Embodiment

It is developed as an embodied agent by its physical components joints, actuators, and a many of sensors that are providing internal stimuli and external stimuli information about the body and its surroundings. Actuation is effected through a variety of DC motors and servo-motors. Sensory data provides joint position, velocity of movements, and torque, streamed video images from the head eye cameras, audio from the left and right ear microphones[**9**]. Accessing to the motor control and sensor interface is provided through iCubInterface in the cognitive architecture in Fig. 5.1 This module is represented as an YARP software module iCubInterface. That interface turns relevant data into actions.

4.3. Software Representation

Software implementation is shown in Figure 5.2 has been realized as a complete software system consist of a YARP and iCub modules. All modules are connected internally and communicates through YARP ports and they can be called from a application or from command line interface with required paramaters. These modules are:

- salience
- endogenousSalience
- egoSphere
- attentionSelection
- episodicMemory
- proceduralMemory
- affectiveState
- actionSelection
- controlGaze2
- logPolarTransform
- cameraCalib

4.4. Gaze Control

The Gaze Control component provides coordinated control of the head and eyes of the iCub. Rather than by specifying the raw joint values for the head and eyes, the gaze direction of the robot is specified by gaze direction (azimuth and elevation angles) and vergence of the two eyes. In addition, the motion of the eyes and head when moving to a given gaze position are controlled to provide a motion profile that is similar to humans, with the gaze direction of the eyes moving quickly and the head moving more slowly but subsequently catching up so that the eyes gaze is reaching an equilibrium position that is relatively centred with respect to the head and with the head oriented in the specified gaze direction. The gaze direction can be specified using

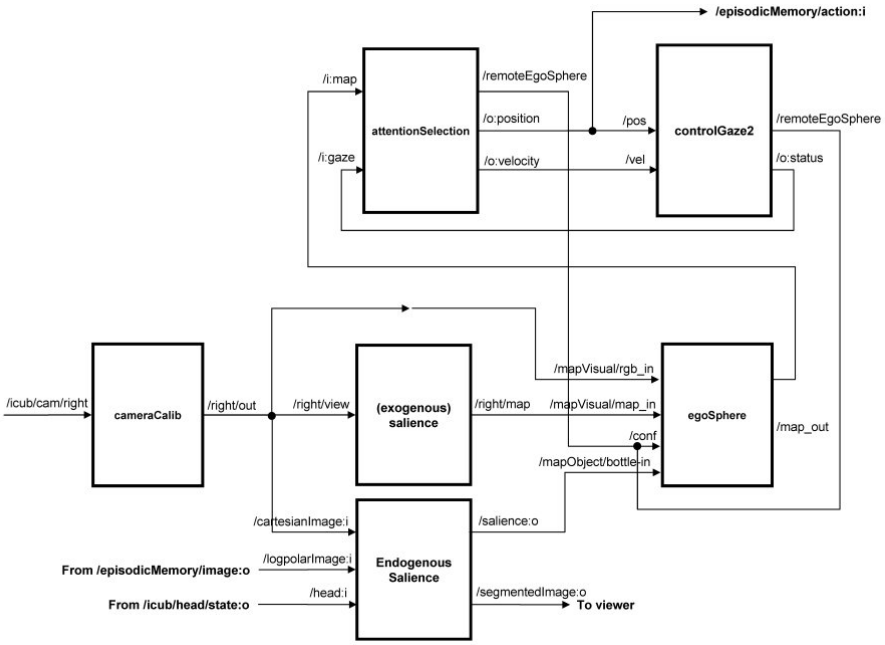

Figure 4.2. iCub Internal Architecture

any of four different types of coordi- nates:

- absolute azimuth and elevation in degrees;
- azimuth and elevation angles relative to the current gaze direction;
- normalized image coordinates (in the range zero to one) of the position at which the iCub should look;
- the image pixel coordinates of the position at which the iCub should look. Head gaze and vergence are controlled simultaneously.

There are two modes of head gaze control: saccadic motion and smooth pursuit motion. Saccades are controlled in two phases. In the first fast phase, the eyes are driven quickly to the destination gaze direction by controlling the common eye ver- sion and tilt degrees of freedom. In the second slow phase, the neck moves toward the final gaze direction with a slower velocity profile and the eyes counter rotate to keep the image stable. A new saccade is accepted only when the previous one has finished. This is the typical operation in humans where saccades are used to change the object of interest. Smooth pursuit only operates in the slow phase, but it accepts a continuous stream of commands.[**1**]It is meant to emulate the human behaviour when tracking an object. A single set of motor controller gains are used for both the saccade and smooth pursuit modes. These gains specify the speed of the motion and therefore the amount of motion undergone by the eyes and by the neck. The higher the gain, the more the eye motion will lead the neck motion and, therefore, the more the eyes have to counter-rotate as the neck approaches the gaze direction. Vergence operates continuously in a single phase and there is an independent gain for the vergence controller.

4.5. Reach and Grasp

The Reach and Grasp includes a robust task-space reaching controller for learning internal generic inverse kinematic models and human-like trajectory generation. This controller takes into account various constraints such as joint limits, obstacles, redundancy and singularities. It also includes a module for grasping based on reaching and orienting behaviours. This allows the coordination of looking (for a potential target), reaching for it (placing the hand close to the target) and attempting a grasping motion (or another basic action). At present, the reaching controller takes as input the Cartesian position and ori- entation of an object to be grasped, based on a visual recognition process. The arm and torso configuration to achieve the desired pose (i.e. the hand position and ori- entation) is determined using a non-linear optimizer which takes into account all the various constraints, one of which is to keep the torso as close as possible to vertical while reaching. Subsequently, human-like quasi straight hand and arm trajectories are then generated independently using a biologically inspired controller. This controller exploits a multi referential system whereby two minimum-jerk velocity vectors are generated, one in joint space and one in task space. A coherence constraint is imposed which modulates the relative influence of the joint and task space trajectories. The advantage of such a redundant representation of the movement is that a quasi straight line trajectory profile which is similar to human like motion can be generated for the hand in the task space while at the same time retaining convergence and robustness against singularities.

4.6. Action

Guideline 10 states that movements should be organized as actions. Since actions are planned, goal-directed, acts they are triggered by system motives (see Guideline 6) and they are guided by prospection. In other words, the action is defined, not by a servo-motor set point specifying an effector movement, but by the goal of the action, an action whose outcome must be achieved adaptively by constituent movements. This guideline is implemented in the iCub cognitive architecture by the procedural and episodic memories and by use of visual servoing in reaching and locomotion. Specifically, the procedural memory represents actions as gaze saccades together with an optional reaching, hand-pushing, grasping, or locomotion movement. These actions are associated with expected outcomes defined by the expected outcome in the episodic memory. Thus, the procedural and episodic memory provide a feed- forward goal state for the action while the visual servoing, whereby the effector is adaptively controlled to align it with a fixation point while re-centering the gaze after the execution of saccade, achieves the required motion through feedback control of the arm and hand.

4.7. Action Selection

The purpose of the Action Selection component is to effect the development of the iCub and specifically to increase its predictive capability. In the current version of

the iCub cognitive architecture, this means the selection of the iCub's mode of exploration. At present, only two basic motives drive this iCub development, curiousity and experimentation, both of them exploratory. The third main motive, social interaction, has not yet been addressed.[**10**] Consequently, the present implementation of action selection is based on a very trivial function of the levels of curiousity and experimentation produced by the Affective State component. Specifically, the learn- ing mode is selected if the curiosity level is higher than the experimentation level; otherwise the prediction mode is selected.

4.8. The YARP

It is a set of open source libraries that supports modularity by using abstraction method in softwares to handle common difficulties in robotics area which are know as modularity algorithms, hardware interfaces and OS platforms.To deal with OS spesific builds,requires to use cross-platform build tools such as CMake[7] and ACE[8] . YARP is providing platform independence. First abstraction can be described as a protocols. Main YARP protocol manages interprocess communications in operating systems. It can deliver process messages of any size across the network by using different protocols. Second abstraction is about hardware communications.The method is to define

Figure 4.3. YARP Command Line Interface

interface for class of devices to fold native coded APIs.Changes in hardwares requires changes in API[9] calls via linking suitable libraries to encapsulate hardware dependency problems. These abstractions combined to use remote device drives where that can be

[7]CMake : http://www.cmake.org
[8]The ADAPTIVE Communication Environment : http://www.cs.wustl.edu/ schmidt/ACE.html
[9]API : Application Programming Interface

accessed across the network like a parallel processing. The purpose of YARP ports are to move data from threads to threads over the processes.Flow of the data can be configured and observed from command-line at real time. Port can receive or send data from any other port.Connections between ports can be modified easily with using different protocols such as TCP[10] and UDP[11] .The choice depends on quality of message transmission or response time. Using TCP is for reliability and UDP is for speed with effect on unreliable transmissions. As it seen in Figure 5.3 it registers a name server over localhost and assigns root node to branching iCub components.

4.9. iCub Simulation Environment

The computer simulation model of the iCub allows to create realistic scenarios in where robot can interact with a virtual world and physical limitations, interactions that occur between the virtual world is simulated using open source library ODE to provide accurate simulation of body dynamics. Simulation surroundings can be manipulated by a single configuration file which contains ; textures, objects, default arm positions, video configurations and calibration settings. In order to place some objects in the environment, it needs to be spawned by using the command line interface. Environment supports direct frame streaming through yarpview application.

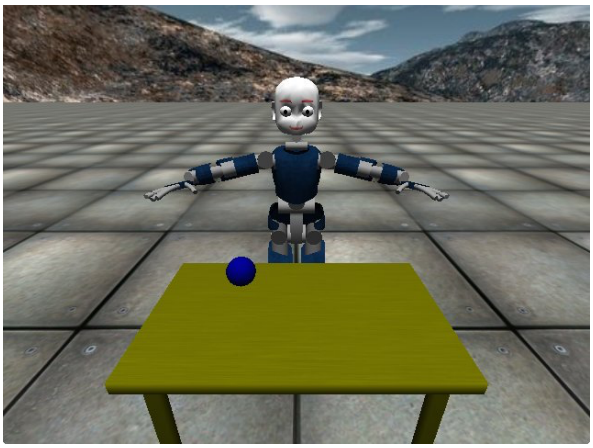

Figure 4.4. iCub Simulation Environment

[10]Transmission Control Protocol
[11]UDP : User Datagram Protocol

5. CONCULUSIONS

5.1. Speech Synthesis

Speech module has ability to speak given text inputs only in English language. A new Module developed from application template with open source text to speech library. iCub now can speak Turkish language. Module can be extended to synthesis of another languages. eSpeak wrapper can provide various of language sound database. Speech Recognition module exists but it needs some Windows system libraries.

5.2. Object Grasping

Module tested with example data sets. It can grasp an object from table. It requires that other modules support. Required modules must be running before the grasping module. It needs to be trained from a neural network file.

5.3. Image Capture

An application developed from templates to capture actual action sequences from head camera. Images can be streamed as a video.

5.4. Sound I/O

Small module developed to send sound commands from computer microphone. Module documentation was not enough descriptive to create new audio ports for speech recognition. Some applications were not complied at all.

All implementations are avaible on **GitHub.**

APPENDIX A: Sample Code Snippets

A.1. Turkish Synthesis Module

```
class iSpeak : protected BufferedPort<Bottle>,
public    RateThread
{
string name;
string package;
string package_options;
deque<Bottle> buffer;
Mutex mutex;
bool speaking;
MouthHandler mouth;
// gets text sequence
void speak(const string &phrase)
{
string command("echo␣\"");
command+=phrase;
command+="\"␣|␣";
command+=package;
command+="␣";
// Check for speech synthesis library
if (package=="espeak")
command+="-v␣turkish␣--stdin";

// Get text input from CLI
command+=package_options;
//int ret=system(command.c_str());
}
void run()
{
```

```
string phrase;
double time;
bool onlyMouth=false;
int rate=(int)mouth.getRate();
bool resetRate=false;
double duration=-1.0;

mutex.lock();
// protecting also the access of size() function
if (buffer.size()>0)
{
// allocate space from yarp Bottle
Bottle request=buffer.front();
buffer.pop_front();

if (request.size()>0){
  if (request.get(0).isString()){
      phrase=request.get(0).asString().c_str();
      speaking=true;
  }
else if (request.get(0).isDouble() || request.get(0).isInt())
  {
time=request.get(0).asDouble();
speaking=true;
onlyMouth=true;
}
}
```

A.2. Object Mover Module

```
public:objectMoverThread(ResourceFinder &_rf) : rf(_rf) {
virtual bool loadParams() {
```

```
name = rf.check("name",Value("objectMover")).asString().c_str();
neckTT = rf.check("neckt t",Value(2.0)).asDouble();
eyeTT = rf.check("eyett",Value(1.2)).asDouble();
trajTime = rf.check("trajtime",Value(4.0),"Solver trajectory
time").asDouble();
maxPitch = rf.check("maxpitch",Value(30.0),"Torso max pitch").asDou

//get which arm to use. default to left if they didnt pass in left
armname = rf.check("arm", Value("left"),"arm name").asString().c_st
if (armname == "right") {
armInUse = true;
}
else {
armInUse = false;
}
//get robot name to use (icubSim is for simulation)
robotname = rf.check("robot", Value("icubSim"),"robot name").asStri
}
```

APPENDIX A: Screenshots

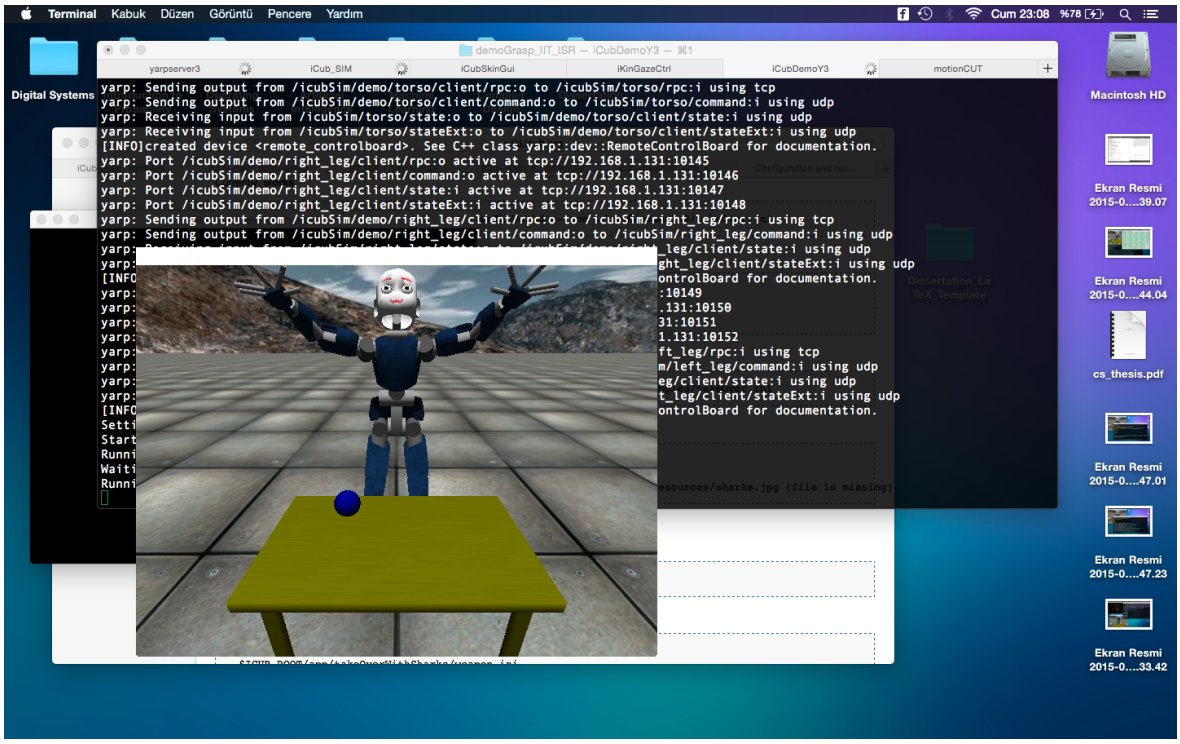

Figure A.1. Full Body Movement

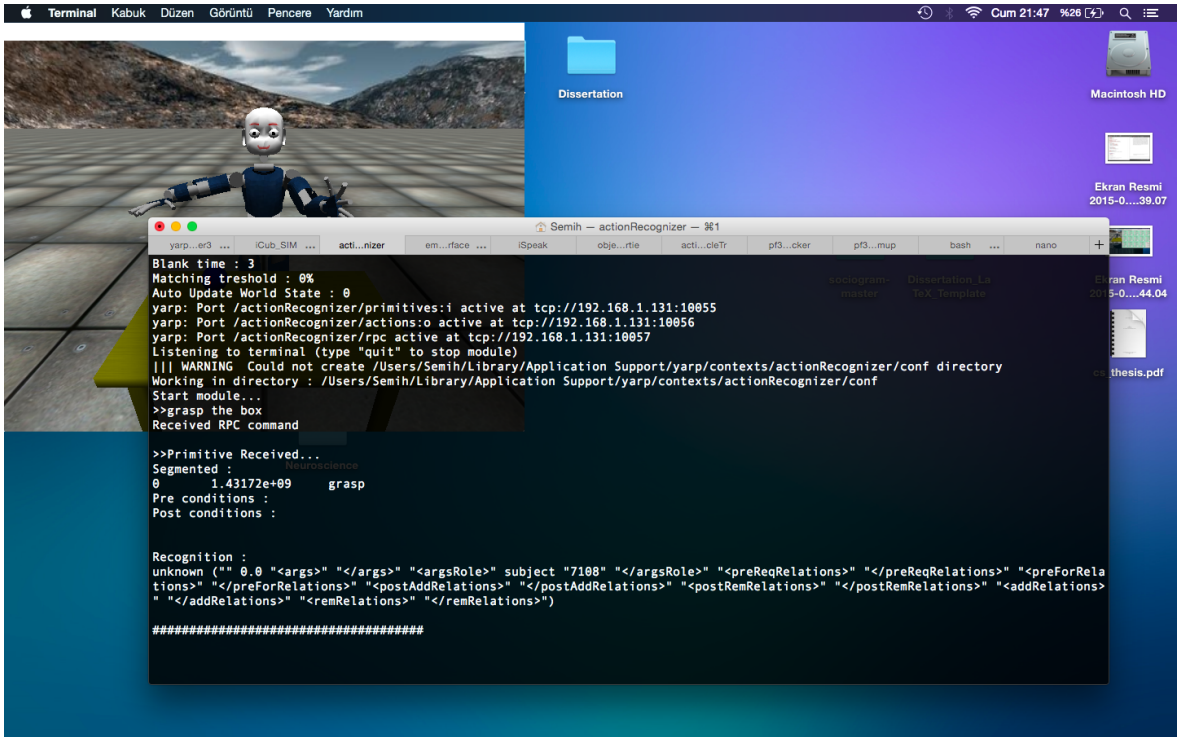

Figure A.2. Action Recognizer

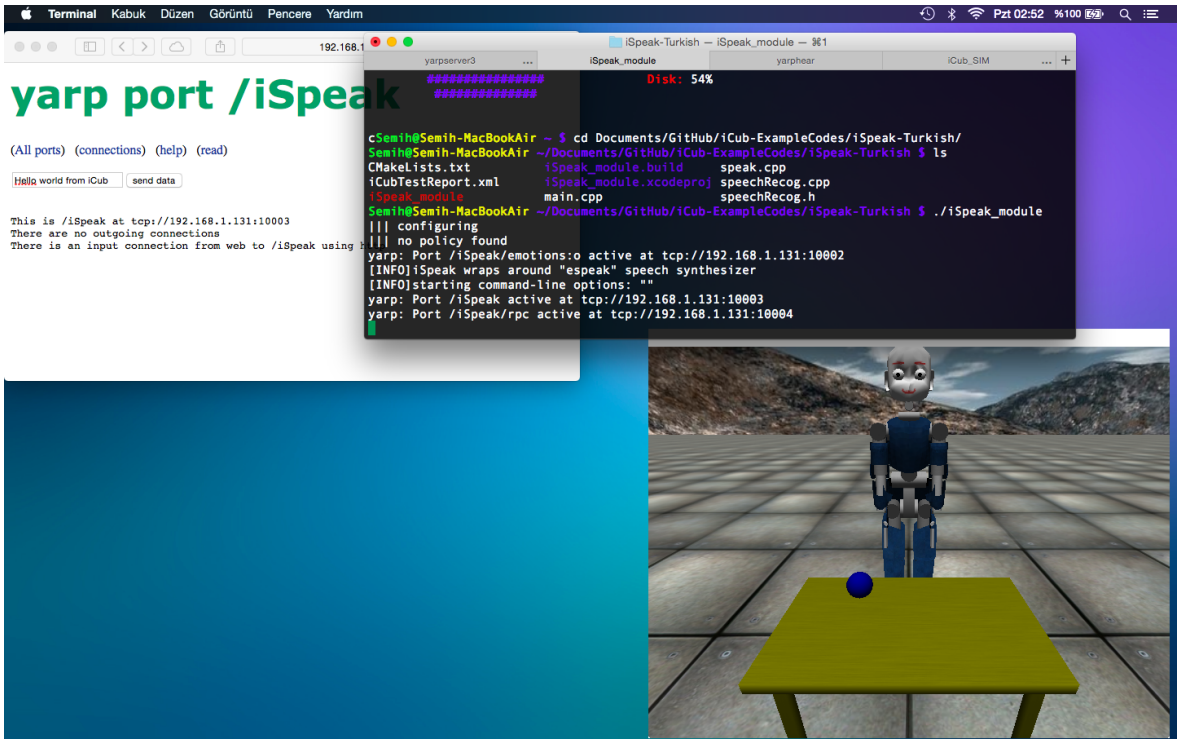

Figure A.3. iSpeak Module

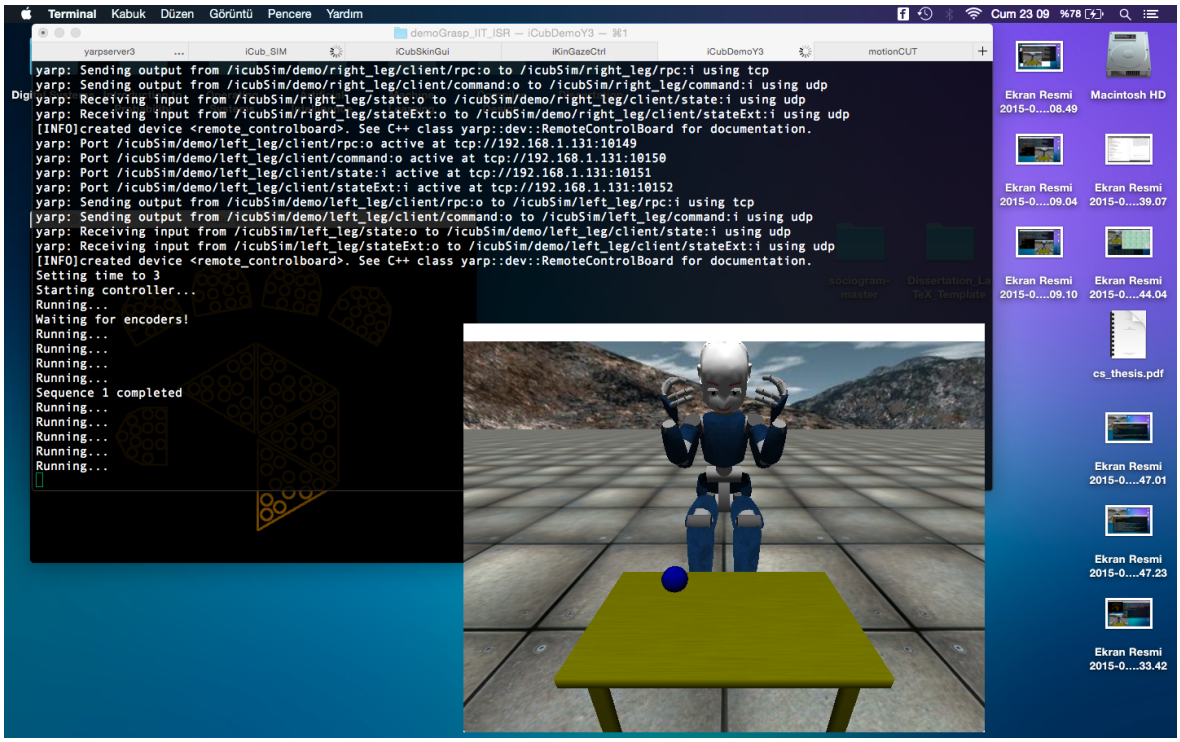

Figure A.4. iCub Sample Body Postures

Bibliography

1. **Webots: Robot simulation software**
   https://www.cyberbotics.com

2. **ODE: Open Dynamics Engine**
   http://www.ode.org

3. **Gazebo: Open Source Simulation Environment**
   http://gazebosim.org

4. **YARP: Yet Another Robot Platform**
   http://wiki.icub.org/yarp/

5. **CMake: Cross-Platform Open Source Build System**
   http://www.cmake.org

6. **ACE: The ADAPTIVE Communication Environment**
   http://www.cs.wustl.edu/ schmidt/ACE.html

7. **iCub: An Open Source Cognitive Humanoid Robotic Platform**
   http://www.icub.org

8. **V. Tikhanoff, A. Cangelosi, G. Metta Platform Speech and Action Integration in Humanoid Robots: Simulation Experiments with iCub, V. Tikhanoff, A. Cangelosi, G. Metta Platform**

9. **The iCub humanoid robot: an open platform for research in embodied cognition, Giorgio Metta Giulio Sandini, David Vernon, Lorenzo Natale Francesco Nori**

10. **Real-Time Parallel Processing of Grammatical Structure in the Fronto-Striatal System: A Recurrent Network Simulation Study Using Reser-**

voir Computing , Xavier Hinaut, Peter Ford Dominey