HUMANOBS I/O Platform Specification

Authors

Helgi Páll Helgason
Hannes Högni Vilhjálmsson
Kristinn R. Thórisson
Eric Nivel

PART 2 OF 2 OF DELIVERABLE D3 Specification of the Experimental Platform					
BELONGS TO WP:	WP 1 – Auto-reconfigurable Architecture				
WP LEAD:	RU-CADIA				
WP PARTICIPANTS:	RU-CADIA				
	ID #	**WP**	**Orig. Date**	**Actual Date**	**REMARKS**
DATA	3	1	Mo6	Mo6	

1. Introduction

This document contains specifications for the HumanObs I/O platform (hereafter platform) including *hardware devices, software modules and data communication*. The target audience consists of those developing other parts of the HumanObs system. The scope of the document is limited to the platform with references to other components of the HumanObs system occurring as applicable. The organization of the document is as follows: First a high-level overview of the platform is given. Secondly, required hardware and software components are listed and explained. Finally, the data communication capabilities of the platform (i.e. communication between the platform and other components), which consist of mBrane messages, are detailed.

2. Overview

The HumanObs I/O platform is the front-end of the HumanObs system and is responsible for monitoring behavior exhibited by the human users as well as presenting an up-to-date view of the virtual environment and all that it contains to the user along with audio (speech).

The following behaviors of the human user are monitored by the platform:

a) Speech (what user says)
Speech recognizer converts what the user says to text.

b) Pitch (how user speaks)
A pitch tracking module continuously tracks the pitch of the user's speech.

c) Gaze (where user looks)
A specialized software module in conjunction with infrared lights and a dedicated infrared camera track the gaze direction of the user.

d) Head movement (how user moves and orients head)
A computer vision module in conjunction with a camera track head movements of the user.

e) Hand movement (how user moves and orients hand)
Data glove worn by the user supplies hand position and orientation.

Maintaining an up-to-date representation of the 3D environment involves:

a) Maintaining position and orientation states for all things in the environment
There are two sources of events that produce changes in physical position and orientation of objects in the 3D environment. It is the responsibility of the

platform to enforce simple physics (gravity, collisions) and reflect the effects of these continuously.

1) Head and hand movements of avatar bodies
2) Physics

b) Producing audio for environment events

This is limited to producing the speech of the communication partner.

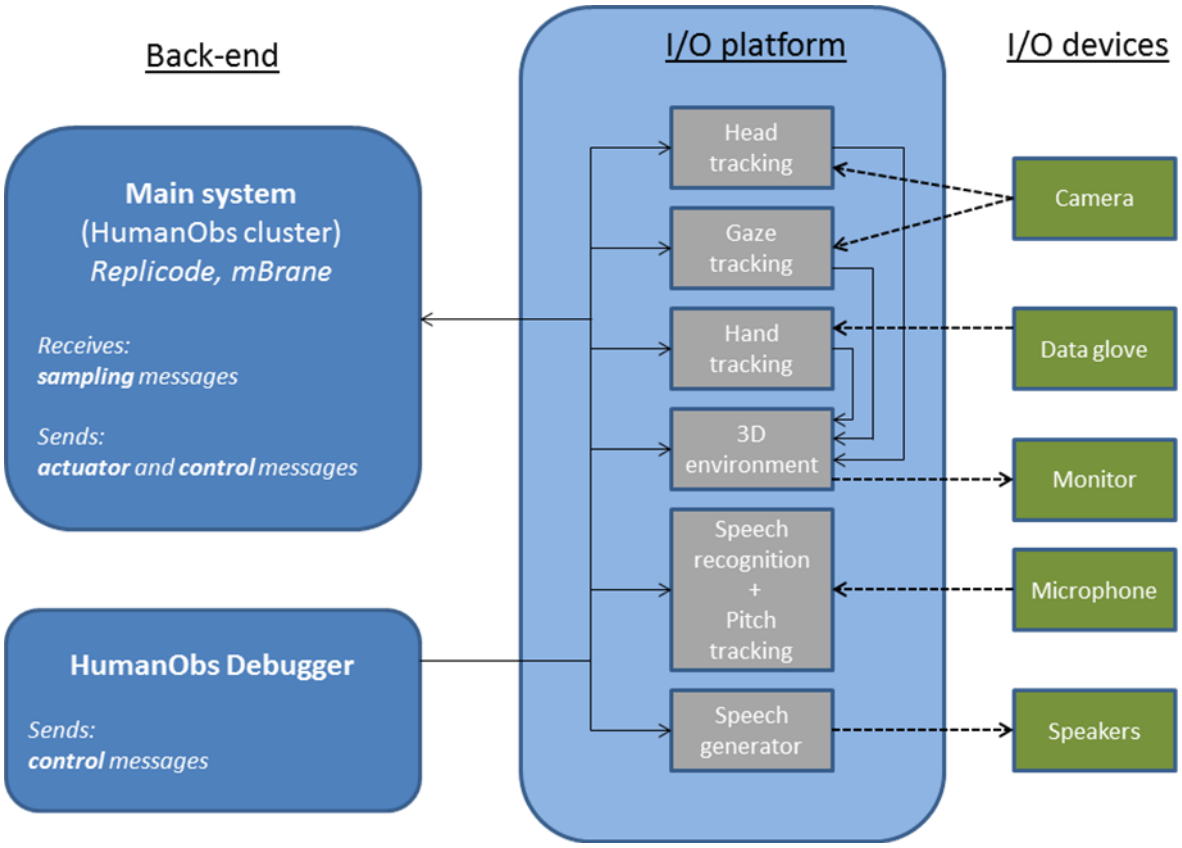

Figure 1: This diagram gives an overview of the platform, components and data communication.

Each grey box in Figure 1 is an mBrane platform module. Where required, there is communication (mBrane messages) between platform modules although specification of these is outside the scope of this document. All platform modules contain translation capabilities for translating to and from Replicode structures.

3. Hardware requirements

a) Machine

The I/O machine shall be equipped with at least a quad core processor running each core at no less than 2.66 gigahertz. At least 2 gigabytes of RAM are required. A

Direct3D 9 compatible display adapter is required. Firewire 400Mbp/s connection is required for gaze tracking. Audio input is required for speech recognition and pitch tracking. Audio output is required to hear generated speech. A USB 2.0 connection is required for the data glove.

b) Head tracking, gaze tracking

Head tracking requires a USB 2.0 or Firewire 400Mbp/s camera with a minimum resolution of 640x480. Gaze tracking requires a Firewire camera with optical zoom and infra-red (night-vision capabilities) capabilities, minimum horizontal resolution of 720 pixels and capture at no less than 24 frames per second. The Sony HDR-HC5 camcorder is recommended. Two infrared lights are also required, Sony HVL-IRM lights are recommended. Head and gaze tracking share the same camera. Any camera meeting the gaze tracking requirements also meets the head tracking requirements.

c) Hand tracking

The P5 VR glove is used for hand tracking.

d) Monitor

An LCD monitor with an aspect ratio of 16:9 or 16:10 and a minimum horizontal resolution of 1600 pixels is recommended.

e) Speech recognition and pitch tracking

An analog head-mounted microphone is required for speech recognition and pitch tracking. A microphone input is required on the I/O machine.

f) Speech generation

Headphones, analog or digital, are required to hear generated speech. A line-out or headphone output is required on the I/O machine.

Summary

I/O machine: quad core CPU @ > =2.66 gigahertz, 2 gigabyte RAM, Direct3D 9 compatible display adapter, Firewire 400Mbp/s (1) & USB 2.0 (2) connections

Optical zoom camera with infrared capabilities, >=720 pixel horizontal resolution and real-time Firewire 400Mbp/s output (Sony HDR-HC5 camcorder is recommended)

Two infrared lights (Sony HLV-IRM are recommended)

P5 VR glove

LCD monitor with 16:9 or 16:10 aspect ratio and >1600 pixel horizontal resolution.

Head-mounted analog microphone and headphones (headset)

4. Software requirements

All binaries of the platform will be supplied by CADIA, including third party libraries that are direct dependencies of the platform. This does not include the operating system (Windows 7), device drivers or redistributable installations.

Third party libraries that are dependencies of the platform are listed below. Installation of these is not required to run the platform. To enable compilation of the platform source code, installation of these is necessary.

Ogre3D (3D graphics)

OpenCV (machine vision)

ITU GazeTracker (gaze tracking)

Microsoft SAPI 5.0 (speech recognition, SDK required for compilation)

Loquendo TTS (speech generation, SDK required for compilation)

The following explains how requirements of the platform are mapped to software components.

a) Head tracking

For head tracking a C++ library developed at CADIA will be used. OpenCV is a dependency of this library.

b) Gaze tracking

Gaze tracking is performed by ITU Gazetracker which is an open source C# application.

c) 3D environment

Rendering the 3D environment is handled by a C++ application developed at CADIA. The open-source framework Ogre3D is a dependency. As the Direct3D mode of Ogre3D is recommended, installation of any 2009 or 2010 version of the Microsoft DirectX redistributable is required.

d) Hand tracking

An inverse kinematics library developed at CADIA will be used to map glove movements to the 3D avatar.

e) Speech recognition

Microsoft SAPI 5.0 is used for speech recognition. It is built in to all flavors of Windows 7.

f) Pitch tracking

Pitch tracking is performed by a C++ library developed at CADIA that analyses the pitch and silences of a real-time, continuous speech, and describes the status of:

- instantaneous pitch and derivative;
- speech on/off;
- speech paused;
- humming;
- speech without a pitch: typically, the analyzer has failed to assign a pitch to the sound, but still, the amplitude is high enough to differentiate it from noise;
- average interrupted speech segment duration;
- average inter-segment duration.

g) Speech generation

Loquendo TTS is used for speech generation.

Summary

Microsoft Windows 7 32-bit (any flavor)

Microsoft DirectX Redistributable (any 2009 or 2010 release)

ITU Gazetracker

Loquendo TTS

CADIA developed C++ libraries for: head tracking, inverse kinematics, pitch tracking, 3D rendering

5. Data communication

All communication between back-end components and the platform consist of mBrane messages. There are three types of messages:

1) Sampling messages

A sampling message consists of sensory data from either the input hardware devices or the 3D environment.

2) Actuator messages

An actuator message is a command to an actuator of the avatar body controlled by the HumanObs main system.

3) Control messages

Control messages are used by the debugger to pause and resume the platform and by the HumanObs main system to control the broadcast frequency of sampling messages.

6. Sampling messages

All sampling messages are sent in streams with each stream having a unique identifier. An additional identifier, "Entity", is contained if applicable in each sampling message and indicates which entity (interviewer or interviewee) the sample applies to.

Entity ID	Description
0	None
1	Interviewer
2	Interviewee

Table 1: Identifiers for sampled entities.

Stream ID	Content description
1001	Head location and orientation
1002	Gaze (identifier for gazed object)
1003	Hand position and orientation
1004	Speech content and pitch (character string, integer array, confidence value)
1005	Position and orientation of non-entity objects

Table 2: Identifiers for sampling data streams.

Each object in the 3D environment has a unique identifier which is used as reference in sampling messages when applicable.

Object ID	Description
0	None
1	Interviewers head (used for gaze only)
2	Interviewers hand (used for gaze only)
3	Interviewees body
4	Interviewers cup
6	Interviewees head (used for gaze only)
7	Interviewees hand (used for gaze only)
8	Interviewees body
9	Interviewees cup
10	Walls
11	Floor
12	Table
13	Interviewers chair
14	Interviewees chair

Table 3: Identifiers for objects contained in the 3D environment.

The number of sampling message mBrane classes is identical to the number of content streams. These are detailed below. Orientation is specified in pitch, roll and yaw.

a) IOSampleHead

Purpose: Provides position and orientation of avatar head.
Parameters:

EntityId (integer) – Indicates which avatar the data applies to
X, Y, Z (double) – Indicates the position of the head
P, R, Y (double) – Indicates the orientation of the head

c) IOSampleGaze

Purpose: Indicates what object in the 3D environment an avatar is gazing at.
Parameters:

EntityId (integer) – Indicates which avatar the data applies to
ObjectId (integer) – Identity of object being gazed at

d) IOSampleHand

Purpose: Provides position and orientation of avatar hand.
Parameters:

EntityId (integer) – Indicates which avatar the data applies to
X, Y, Z (double) – Indicates the position of the hand
P, R, Y (double) – Indicates the orientation of the hand

e) IOSampleSpeechPitch

Purpose: Provides content and pitch data for speech. Speech content is processed at the granularity level of syllables.
Parameters:

EntityId (integer) – Indicates which avatar the data applies to
Content (string) – One syllable as a character string
Pitch (array of integer) – Pitch values for the syllable
Confidence (double) – Value in the range [0..1] indicating confidence

f) IOSampleNonEntityObject

Purpose: Provides position and orientation of a non-entity object.
Parameters:

ObjectID (integer) – Indicates which object the data applies to.
X, Y, Z (double) – Indicates the position of the object
P, R, Y (double) – Indicates the orientation of the object

7. Actuator messages

Actuator messages are sent by the HumanObs main system to the platform in order to control avatar actuators (head, hand, speech). An entity identifier is included, although it is assumed that the main system only controls one avatar at a time. For

motor control actuator commands are issued in terms of desired state (position and orientation). For example, to move the hand from (x1, y1, z1) to (x2, y2, z2) may require sending multiple actuator messages as hand movement is restricted to a certain distance for each rendered frame. The same is true for orientation.

Below is a list of actuator messages supported by the platform:

a) IOActuatorHand

Purpose: Commands the hand of an avatar to move closer towards a specified location and orientation

Parameters:

EntityID (integer) – Indicates the hand of which avatar to move
X, Y, Z (double) – Indicates the desired position of the hand
P, R, Y (double) – Indicates the desired orientation of the hand

***b)* IOActuatorHead**

Purpose: Commands the head of an avatar to be oriented closer to a specified orientation

Parameters:

EntityID (integer) – Indicates the head of which avatar to move
P, R, Y (double) – Indicates the desired orientation of the head

c) IOActuatorSpeech

Purpose: Commands an avatar to speak

Parameters:

EntityID (integer) – Indicates which avatar should speak
Content (string) – Text to be converted to speech
Pitch (integer) – Pitch of speech

8. Control messages

Control messages are used for controlling certain operational aspects of platform modules (grey boxes in Figure 1). Precisely, these are: pause, resume and set broadcast frequency. All of these are to be understood in context of the broadcasting platform modules perform. The pause command instructs a platform module to temporarily suspend its broadcasting (it stops sending sample messages). The resume command instructs a module in a paused state to resume broadcasting (it will again send sample messages). The set broadcast frequency command instructs a platform module of the frequency at which it should send sample messages with the frequency being given as microsecond intervals.

Note that control commands only apply to the messaging operation of platform modules. Internally, platform modules always sample at their native temporal resolution.

To enable correct routing of control messages, platform modules are given a unique identifier. Note that the Speech generation module is a special case and receives no control messages as it performs no sampling (it is an output-only module).

Module ID	Description
0	All modules
1	Head tracking module
2	Gaze tracking module
3	Hand tracking module
4	3D environment module
5	Speech recognition + Pitch tracking module

Table 4: Identifiers for platform modules (grey boxes in Figure 1).

Below is a list of control messages supported by the platform:

a) IOControlPause

Purpose: Instructs a platform module to suspend broadcasting of sample data

Parameters:

ModuleID (integer) – Indicates which module the command applies to

b) IOControlResume

Purpose: Instructs a platform module to resume broadcasting of sample data

Parameters:

ModuleID (integer) – Indicates which module the command applies to

c) IOControlSetFrequency

Purpose: Instructs a platform module to broadcast sample data at the specified frequency

Parameters:

ModuleID (integer) – Indicates which module the command applies to

Frequency (integer) – Interval in microseconds between sent samples