Indriya : A platform for designing human robot interaction scenarios based on human behaviors

Praveenkumar Vasudevan[1] and Gentiane Venture[2]

***Abstract*— In scenarios where robots coexist with humans in a social environment, understanding both verbal and non-verbal communication is extremely inevitable. Together they convey information such as intention, emotion and even health of a human, that adds value to the way robots participate in an interaction. On the other hand, the people who design interaction scenarios are from diverse fields who do not essentially have the required robot programming skills. In this paper a new behavior programming paradigm and an easy to use visual programming interface for the same is proposed, which gives the power to design human-robot interaction taking into account human behaviors. The distributed platform namely Indriya proposed in this paper gives the ability to plug and play multimodal sensors and diverse class of robots. The platform usage has been demonstrated using scenarios where NAO robot performs actions understanding human behavior using Microsoft Kinect sensor.**

I. INTRODUCTION

The richness and diversity of Human Robot Interaction (HRI) has been described in [1] as "HRI is a challenging research field at the intersection of psychology, cognitive science, social sciences, artificial intelligence, computer science, robotics, engineering and human-computer interaction". Goodrich in his extensive survey [2] proposed two main types of HRI namely remote interaction and proximate interaction. The latter has led to the development of a new class of robots called social robots. Yan et al. [3] define "A social robot is a robot which can execute designated tasks and the necessary condition turning a robot into a social robot is the ability to interact with humans by adhering to certain social cues and rules."

Social robots already entered the human spaces as entertainers, educators, caring agents and personal assistants [4]. Hence the design and the development of interaction systems need to be approached in a systematic manner wherein the robots should be able to understand the human behaviors in order to interact in a better way. To make it possible it is necessary to develop robotic systems with essential perceptual ability for efficient and natural interaction. Most often the on-board sensors on the robots fail to satisfy this demanding requirement due to various constraints like space, power and computational needs. Therefore consideration of augmenting exteroceptive sensors that are commonly available in the smart home/public environments to this purpose is essential.

Another important aspect is that the HRI designers are from diverse backgrounds. People study various aspects such as robot ethics, social acceptance, liveliness, cultural influence etc., from various perspectives like sociology, psychology, humanities and so on. So the tools needed to design behaviors of a social robot should be intuitive and user friendly. With increased availability of social robots and cost effective human activity recognition sensors, we could still observe a huge void which inhibits the exploitation of available technology for designing robot behaviors for human-in-the-loop scenarios.

The main contribution of this paper are: **(a)** A distributed and modular *Application framework* which gives opportunity to interface multimodal sensor systems and diverse class of robots. **(b)** A simple and easy to understand *Behavior program model* in order to design reactive human robot interaction scenarios. **(c)** An intuitive and easy to use *User interface* to design, execute and monitor interaction from broad range of client devices.

The paper is organized as follows. Section II discusses the state-of-the-art techniques. Section III presents a brief introduction to Indriya architecture. Section V describes the system capabilities and Section VI presents the user study results. Finally Section VII presents the concluding remarks.

II. RELATED WORK

Human behavior understanding can be broadly classified into non-verbal/motion recognition and verbal/speech recognition. Vision based motion capture and analysis systems have been one of the first class citizens in the human motion analysis which is summarized in surveys [5][6]. Vision based human pose estimation has traditionally suffered from the requirement to adopt an initialization pose and losing track after a few frames. These problems have been addressed in [7] which proposes methods to track accurately the human skeletons using single depth images. Understanding of human motion is not complete if the action of the human could not be inferred. In the survey [8], a study on various algorithms used for human activity analysis is presented. Recently data-driven machine learning approaches have proven to be successful with recognition accuracy as high as 94.9% [9]. The verbal communication instead has been studied by human computer interaction community for many years now [10] [11] and led to the development of software toolkits that implements state-of-the-art algorithms [12].

*This work was supported by JASSO

[1]Praveenkumar Vasudevan is a graduate student in robotics at Ecole centrale de Nantes, 1 rue de la noe, 44000 Nantes, France praveenv4k@gmail.com

[2]Gentiane Venture is an Associate Professor at the Department of Mechanical Systems Engineering, Tokyo University of Agriculture and Technology, 2-24-16 Koganei, Tokyo - 1848588, Japan venture@cc.tuat.ac.jp

The localization of humanoid robots is a challenging issue, due to rough odometry estimation, noisy onboard sensing, and the swaying motion caused by walking [13]. Studies on robot localization, obstacle mapping, and path planning by equipping NAO with a consumer-level depth camera have been reported in [14]. Localization and motion planning in smart home environment using an external kinect sensor have been proposed in [13]. These methods are computationally demanding and it could cause overall performance degradation particularly when one wants to share the same sensor for both human motion recognition and localization of the robot. Tracking rectangular fiducial markers using augmented reality tool-kits like ALVAR [15] can be interesting if one could embed those markers on the humanoid robot. This is one of the simplest and cheapest solutions in terms of computational power as it can provide position and orientation of the physical markers in real time.

The users of social robots do not have necessarily backgrounds in programming and design of robot behaviors. The main challenge in the behavior design is the ability to define the behavior which can abstract complex data flows from the end user. There exists flow-chart based visual programming languages [24] which allow non-programmers to create robot applications using a set of pre-built behavioral blocks. These programs are very intuitive but when it comes to designing reactive behaviors for human-in-the-loop scenarios, the existing visual programming methods increase the cognitive load on the end users. Specialized robot programming techniques like Task description language [17] and middlewares like ROS [18] have been proposed in the literature. Though these systems provide modular and distributed architecture, support multiple sensors and robots etc., these require high level of skill in robotics and programming. Recently non-domain-specific solution like Targets-Drives-Means is proposed in [19], however it lacks an intuitive interface.

III. INDRIYA PLATFORM

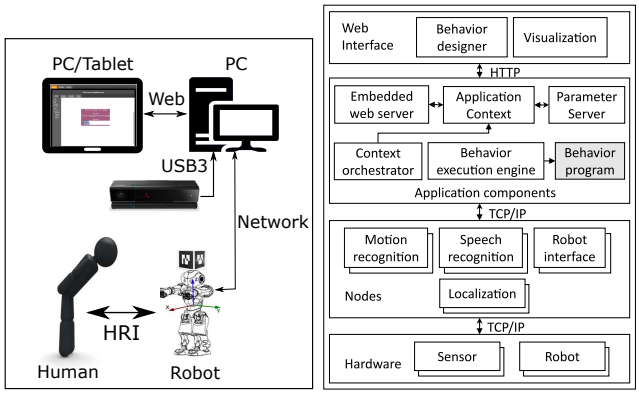
We propose a platform namely "Indriya", for designing human robot interaction scenarios taking inspiration from distributed architecture [18] and intuitive visual programming techniques [20].

A. System Architecture

The system setup and architecture are shown in Fig 1. The principal components of the architecture are

1) Hardware layer: : It is composed of robots and sensors which are connected to the Indriya application server.

2) Distributed Layer: : It is composed nodes each with a specific goal that can run in any machine inside the network. All the nodes will communicate with the application using message passing techniques. The most important nodes in the system are: *Motion Recognition Node* that interacts with a motion recognition sensor and sends the detected motions and gestures to the application. Additionally each motion recognition module registers a set of motions/gestures that could be detected with the sensor associated with it, *Speech Recognition Node* that makes use of the pre-configured



(a) System setup (b) Software architecture

Fig. 1: Indriya Platform

grammar file for language of interest to recognize the human speech, *Robot Interface Node* that interacts with a specific robot and can invoke a set of actions on it. It also sends periodic update about the robot status to the application. Moreover it registers a set of parameterizable actions that could be invoked on the robot associated with it.*Localization Node* that uses the perception system to compute the position and orientation of the robot and humans in the environment.

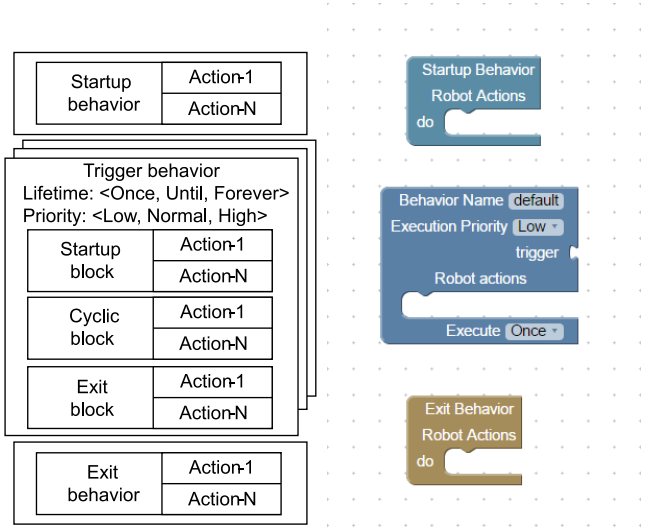
3) Application Components: The *Application context* contains the complete description of the world. It contains latest information about all the robots and humans in the environment. The *Parameter Server* acts as a central repository for managing the parameters of the system. The *Embedded Web Server* is responsible for data and file transaction between the Indriya system and the web client through a set of RESTful services. The *Context Orchestrator* keeps the application context uptodate by synchronizing with information published by the distributed components.

4) User Interface: : The user interface is a web application that runs on any latest web-kit browsers supporting WebGL technology. The prime goal of this UI is to make it suitable for environments adopting bring your own device (BYOD) policy. The UI is composed of: A *Behavior Designer* that could be used by the user to drag and drop the behavior blocks and construct the program by putting together motion recognition blocks and robot action blocks. The designer offers a full range of capabilities like Create/Edit/Delete/Save behavior programs. The *Visualization* could be used to see the interaction of the human and robot inside a virtual 3D environment.

B. Behavior Program

The behavior program is structured in a simple way so that it could be easily understood by the end user. The conceptual model of behavior program is shown in Fig. 2a and the block level implementation is shown in Fig. 2b. The behavior program is composed of:

Startup and Exit behavior: The start-up behavior will be executed once when the user starts the program. The user can add a set of actions to be performed when the program



(a) Conceptual Model (b) Block Implementation

Fig. 2: Behavior program structure

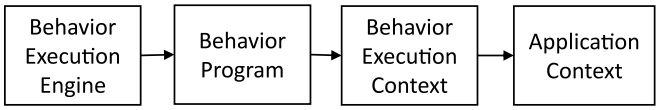


Fig. 3: Behavior program - Execution flow

starts. Similarly the exit block will be executed once when the lifetime of all the configured behavior blocks expire. Both these blocks are optional and there cannot be more than one instances of these in a program.

Trigger behavior: The behavior block is the core component of the behavior program. The block could be activated by the interaction between the human and the robot. There could be many behavior blocks in a program.

TABLE I: Trigger behavior properties

Property	Description
Trigger	Block activation trigger. Gesture/speech trigger, vicinity of human etc.,
Lifetime	Behavior lifetime. Once, Forever or Until a condition is met
Priority	Behavior priority. Uses fixed priority premptive scheduling

C. Code generation and execution

Each of the visual behavior program blocks has an associated code generation module and C# code is generated from the blocks from the blocks. Traditionally there is a need to compile the C# code and generate binaries when one wants to execute it. Thanks to the latest compiler technologies and tools like ScriptCS [21], it is now possible to run the C# code like scripts. This is exploited in the Indriya platform.

The execution flow of the behavior program is shown in Fig. 3. The generated behavior program is consumed by a *Behavior execution engine* which dynamically identifies the

Algorithm 1: Behavior execution algorithm

```
Data: program : Behavior program
1  startup := STARTUP_BEHAVIORS(program);
2  exit := EXIT_BEHAVIORS(program);
3  triggered := TRIGGER_BEHAVIORS(program);
4  sorted := SORT_PRIORITY(triggered);
5  if startup then
6      EXECUTE_BEHAVIOR(startup);
7  while behavior lifetime do
8      forall the behavior in sorted do
9          if behavior not complete then
10             if CHECK_TRIGGER(behavior) then
11                 current = GET_ACTIVE_TASK();
12                 priority1 = PRIORITY(behavior);
13                 priority2 = PRIORITY(current);
14                 if priority1 > priority2 then
15                     DO_PREMEPTION(current);
16                     SCHEDULE_TASK(behavior);
17                 else if no active task then
18                     SCHEDULE_TASK(behavior);
19 if exit then
20     EXECUTE_BEHAVIOR(exit);
```

configured behavior blocks. The main functionality of the behavior execution engine is to check if the trigger condition of the behavior blocks is met and once the condition is met, the execution of the corresponding behavior is scheduled on a separate CPU thread. The behavior program accesses the uptodate information about the environment through an interface namely the *Behavior execution context* which acts as a proxy to the *Application context*. The algorithm of the behavior execution by the engine is shown in Algorithm 1.

IV. IMPLEMENTATION

The platform is implemented using open network communication standards equipped by ZeroMQ library [22] and message serialization is done using Google protocol buffers [23]. The node information and parameters of nodes are configured using an XML file. A set of human motion clips are captured and gesture recognizer is trained using the Visual gesture builder tool that comes with Microsoft Kinect SDK. Similarly a set of commonly used speech commands are configured in the grammar files to be used by the speech recognition node. The localization of the robot is achieved using a marker cube mounted on the head of the humanoid robot. ALVAR [15] marker tracking library is used along with a 2-DOF kinematic model from torso to the tip of the head of the robot. A set of actions are developed for NAO humanoid robot such as approaching a human, saying expressively etc., and corresponding visual blocks are also developed. The software is tested on a 64-bit Intel Core i7 CPU with clock speed 3.60 Ghz and 8 GB of RAM on a Microsoft Windows 8.1 OS. The source

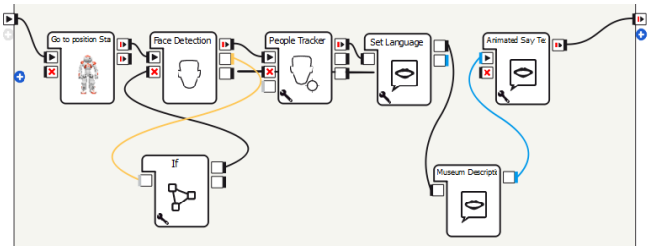


Fig. 4: NAO museum guide: Choregraphe program

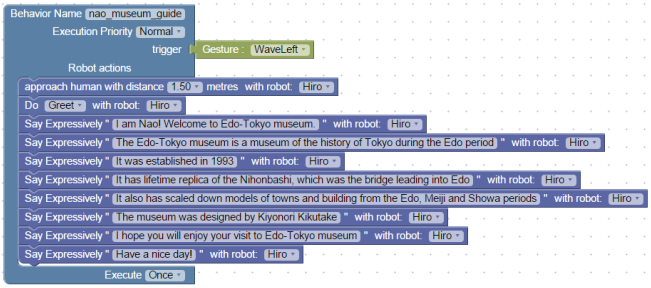


Fig. 5: NAO museum guide: Indriya program

code of the software is available as open source at `https://github.com/praveenv4k/Indriya`

V. SYSTEM EVALUATION

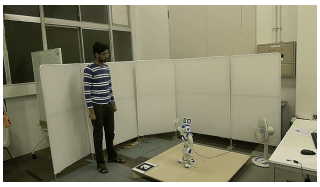
In this section various capabilities of Indriya platform are evaluated

A. Intuitiveness

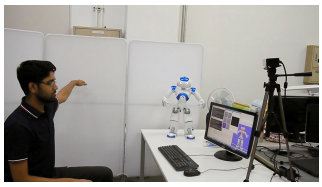
Scenario: The NAO humanoid robot is a guide in a museum. The museum manager would like to design a scenario where when a visitor comes into the vicinity of the robot, the robot would approach him/her and start explaining the history of the museum. The experiment setup for this scenario is shown in Fig 6a.

We would like to use this scenario to compare the simplicity and straight forwardness of behavior description with Choregraphe [24] shown in Fig 4 and with our interface shown in Fig 5. Though Choregraphe uses a familiar flow-chart based programming model and has a huge library of primitive blocks to build complex motion patterns, the data flow for this scenario is not straight forward. As could be noticed from Fig 4, at first the robot keeps looking for people in its vicinity using the *Face detection block* at the cost of its power. Once it detects a person, the face detection block is stopped and *People tracker* block is activated. The robot starts approaching (tracking) the person until a fixed distance with the person is reached. Once the desired distance of separation is achieved, the people tracker block is stopped. Now the robot will actually start explaining the history of the museum. For a novice programmer it could be difficult to think of this dataflow as they literally have to emulate the whole scenario in their mind before designing the system.

The Indriya platform equipped with Kinect sensor takes care of the detection of people and gives the relative localization of the robot and human. Once the person is detected



(a) NAO museum guide



(b) NAO therapy facilitator

Fig. 6: Experiment setup

or a configured gesture trigger arrives, the behavior program retrieves the relative transformation of the robot with respect to human. The *Approach block* makes use of this information to drive the robot towards the person. After coming into the proximity of the person, the robot starts explaining the history of museum. From the user perspective, the design of the behavior is *intuitive* using Indriya. He/She can focus on the scenario rather than thinking about the minute details of the data flow.

B. Realistic scenarios

Scenario: A physiotherapist who is in a remote hospital would like to prepare an exercise routine for his patient who is recovering from the fracture of his left hand. The therapist wants the service robot in the rehabilitation center to give directions to the patient in an interactive manner and facilitate the process. The exercise is composed of: an introduction and demonstration of the routine, interactively reporting the progress of the exercise and finally notifying the completion. The experiment setup is shown in Fig. 6b

Though Choregraphe software gives the capability of programming behaviors with basic human awareness, it does not explicitly allows programming that takes into account human gestures/motions. This is not the drawback of the system but the fact that it does not have enough sensors to understand complex motions. Thanks to the Kinect sensor and its gesture recognition capabilites, one can exploit it to design a program for the above scenario using our platform as shown in Fig 7.

The scenario is designed using all the three basic behavior constructs that constitutes the behavior program. In the startup behavior the robot gives an introduction about the exercise routine and then gives a demonstration of how to do it. A trigger behavior block is configured to be triggered each time when the patient performs the therapy routine (lifting left hand). Each time the patient performs the exercise a variable is incremented and the robot notifies the progress by announcing how many times the patient has completed the exercise. The trigger behavior block is configured *Until* a desired condition on the exercise count is reached (say lifting left hand 5 times). Once the lifetime of the trigger behavior block expires, the robot gives some closing comments about the routine. It could be noticed that the description of this scenario is quite simple using Indriya. The blockly editor powered with the behavior program logic abstracts the complexity of designing realistic scenarios.

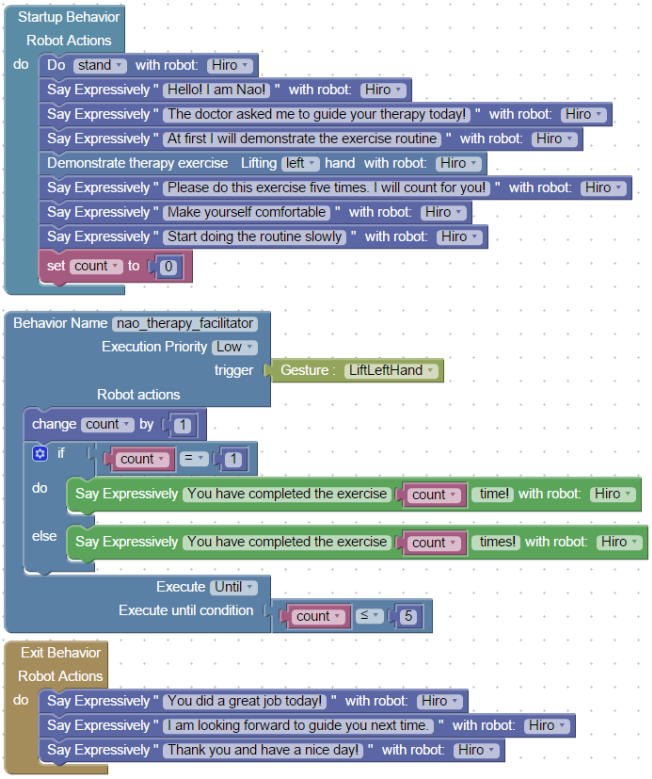


Fig. 7: NAO therapy facilitator: Indriya program

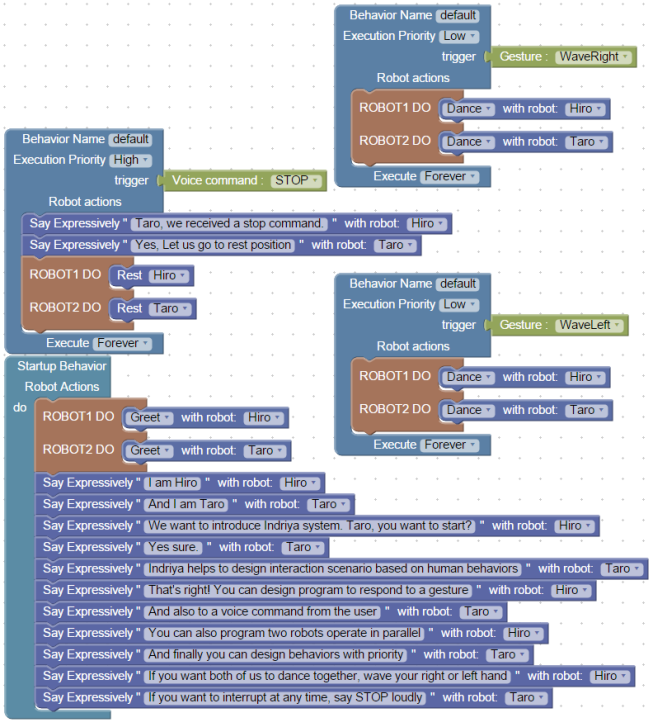


Fig. 8: Product demo scenario: Indriya program

C. Priority execution

Most often in real life HRI scenarios one would like to have robot respond to certain events with a higher priority irrespective of what it is doing at that time. Priority based execution capability comes handy in such scenarios. The inherent problem is that often these concepts are extremely difficult to understand/implement for beginner to intermediate level programmers. The Indriya system makes this extremely easy by just fixing the priority of each behavior blocks.

Scenario: Let us consider an elderly care robot who takes care of a fictitious person Mr. Adams. The living room is equipped with kinect sensor so that the activities and voice commands of the person could be recognized by Indriya system. The care taker of Mr. Adams who is working during the day would like to design a behavior program for the robot. He/She wants the robot to respond to normal commands like *Come here*, *Bring Coffee* etc., Among others he would like the robot to serve emergency conditions like when the elderly person faint down suddenly, or he/she asks for help due to medical complications etc.,

The reference implementation of the caring robot is shown in Fig.

D. Programming Multi-robots and Parallel tasks

The Indriya system is designed in such a way that it is easy to create multiple instances of robot interface module targeted at a single robot type (say NAO) by just changing the connection parameters. The visual blocks are also designed so that it is easy to select the robot on which the task should be executed, by an user-defined name. Additionally the platform makes it simpler to execute parallel tasks on two or more robots without any hassles.

Scenario: Let us consider a product introduction event in which the product team decided to use robots to give an introduction and demonstration of their product. Let us assume that the robots say *Hiro* and *Taro* were used to give an introduction about *Indriya* platform and its capabilities like multi-robot support, parallel execution, gesture/voice recognition support and priority based execution support etc., A reference implementation of this scenario is shown in Fig 8

Even for an intermediate programmer implementing such a scenario using say a system like ROS [18] will be a daunting job. Parallel programming using multiple threads is tricky at times. Depending on the OS and programming language used, the constructs to build programs for parallel execution of tasks is difficult. With robots it is even more difficult job as the execution has to be monitored while serving other high priority human triggers. The Indriya platform gives a breezy user interface with which all these problems are effectively addressed.

VI. USER STUDY

VII. CONCLUSIONS

ACKNOWLEDGMENT

We would like to thank the members of GVLab at Tokyo University of Agriculture and Technology for helping us arranging the resources and participating in the experimentation.

REFERENCES

[1] K. Dautenhahn, "Methodology and themes of human-robot interaction: a growing research field," *Int. J. of Advanced Robotic Systems*, vol. 4, no. 1, pp. 103–108, 2007.

[2] M. A. Goodrich and A. C. Schultz, "Human-robot interaction: a survey," *Foundations and trends in human-computer interaction*, vol. 1, no. 3, pp. 203–275, 2007.

[3] H. Yan, M. H. Ang Jr, and A. N. Poo, "A survey on perception methods for human–robot interaction in social robots," *Int. J. of Social Robotics*, vol. 6, no. 1, pp. 85–119, 2014.

[4] Aldebaran, https://www.aldebaran.com/en, accessed: 2014-11-15.

[5] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer vision and image understanding*, vol. 104, no. 2, pp. 90–126, 2006.

[6] R. Poppe, "Vision-based human motion analysis: An overview," *Computer vision and image understanding*, vol. 108, no. 1, pp. 4–18, 2007.

[7] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake, "Efficient human pose estimation from single depth images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 12, pp. 2821–2840, 2013.

[8] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with Microsoft Kinect sensor: A review," *Cybernetics, IEEE Transactions on*, vol. 43, no. 5, pp. 1318–1334, 2013.

[9] Microsoft, "Kinect for Windows," https://www.microsoft.com/en-us/kinectforwindows/, accessed: 2014-11-20.

[10] D. R. Reddy, "Speech recognition by machine: A review," *Proceedings of the IEEE*, vol. 64, no. 4, pp. 501–531, 1976.

[11] R. P. Lippmann, "Speech recognition by machines and humans," *Speech communication*, vol. 22, no. 1, pp. 1–15, 1997.

[12] *Microsoft Speech Platform*, https://msdn.microsoft.com/en-us/library/dd266409(v=office.14).aspx (Accessed: 2015-05-25).

[13] E. Cervera, A. A. Moughlbay, and P. Martinet, "Localization and navigation of an assistive humanoid robot in a smart environment," in *IEEE Int. Workshop on Assistance and Service Robotics in a Human Environment*. Oct, 2012.

[14] D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3D environments based on depth camera data," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS Int. Conf. on*. IEEE, 2012, pp. 692–697.

[15] *ALVAR : A Library for Virtual and Augmented Reality*, www.vtt.fi/multimedia/alvar.html (Accessed: 2015-02-20).

[16] *Aldebaran NAO Humanoid Robot*, https://www.aldebaran.com/en/humanoid-robot/nao-robot-working (Accessed: 2014-11-15).

[17] R. Simmons and D. Apfelbaum, "A task description language for robot control," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ Int. Conf. on*, vol. 3. IEEE, 1998, pp. 1931–1937.

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," *ICRA workshop on open source software*, vol. 3, no. 3.2, p. 5, 2009.

[19] V. Berenz and K. Suzuki, "Targets-drives-means: A declarative approach to dynamic behavior specification with higher usability," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 545–555, 2014.

[20] *Google Blockly*, https://developers.google.com/blockly/ (Accessed: 2015-04-31).

[21] *ScriptCS*, http://scriptcs.net/ (Accessed: 2015-05-10).

[22] *ZeroMQ*, http://zeromq.org/ (Accessed: 2015-02-25).

[23] *Google Protocol Buffers*, https://developers.google.com/protocol-buffers/ (Accessed: 2015-03-05).

[24] *Aldebaran Choregraphe software*, https://www.aldebaran.com/en/robotics-solutions/robot-software/development (Accessed: 2014-11-20).