Indriya: An intuitive interface for designing social behaviors based on human motions

Praveenkumar Vasudevan[1] and Gentiane Venture[2]

[1] École Centrale de Nantes, France,
praveenv4k@gmail.com
[2] Tokyo University of Agriculture and Technology, Japan
venture@cc.tuat.ac.jp

Abstract. Humans interacting with intelligent robots has been seen as a potential game changer of the future. In scenarios where robots coexist with humans in a social environment, understanding not only verbal communication, but also non-verbal communication is extremely inevitable. The non-verbal communication carries information such as intention, emotion and health of a human, that adds value to the way robots participate in an interaction. Additionally, the people who design interaction scenarios are from diverse fields who do not essentially have the required robot programming skills. In this paper we propose an easy to use and intuitive programming interface which gives the power to design robot behaviors taking into account human motions. We propose a distributed system namely Indriya which gives the capability to plug and play multi-modal motion recognition systems and diverse class of robots. We present results of NAO humanoid robot performing actions understanding human motions using Kinect motion recognition system.

Keywords: human robot interaction, motion recognition, robot behaviors, kinect, nao, motion driven behaviors

1 Introduction

The richness and diversity of Human Robot Interaction (HRI) has been described in [6] 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 [7] proposed two main types of HRI namely remote interaction and proximate interaction. The latter is particularly important where the humans and the robots are co-located. Proximate interaction has gained importance due to the successful encounters of putting robots to work with human beings. It has led to the development of a new class of robots called social robots. Yan et al. [21] 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 [2]. 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 motions and intentions 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 in HRI is the fact that the users of such systems 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 motion recognition sensors, we could still observe a huge void which inhibits the exploitation of available technology for designing human motion driven robot behaviors.

The main contribution of this work is an application independent experimental platform: Indriya, wherein a social robot is augmented with essential perceptual ability to understand human motions. The behavior design of such a social robot is made possible by an easy to use behavior design interface. The resulting experimental platform could be used by people from interdisciplinary fields to design the interaction between social robots and humans. Although our behavior design framework could accommodate variety of robots and sensors, we present the scenarios targeted for the NAO [3] humanoid robot and commercially available Kinect [1] RGB-D camera for localization and gesture recognition.

2 Related work

Vision based motion capture and analysis systems have been one of the first class citizens in the human motion capture and analysis. It has been studied widely and a summary of all the approaches developed during the past decade has been presented in the surveys [14][16]. 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 by the approaches proposed by Xbox team [1] which are capable of accurately tracking human skeletons using single depth images [19]. Understanding of human motion is not complete if the action of the human could not be inferred. In the survey by Microsoft research team [10], 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% [1].

The localization of humanoid robots is a challenging issue, due to rough odometry estimation, noisy onboard sensing, and the swaying motion caused by walking [5]. The Point cloud library [18], one of the most widely used 3d perception library, implements ready to use probabilistic tracking algorithms. Studies on robot localization, obstacle mapping, and path planning by equipping NAO

with a consumer-level depth camera have been reported in [13]. Localization and motion planning in smart home environment using an external kinect sensor have been proposed in [5]. 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 toolkits 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 [3] which allow non-programmers to create robot applications using a set of prebuilt 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 [20] and middlewares like ROS [17] 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 [4], however it lacks an intuitive interface.

Nowadays there has been a lot of efforts to teach programming to children and people without computer science background [12][8]. These tools are very intuitive and have already been proven to be used by novice programmers to build games and educational applications. The Blockly library [8] from Google offers a complete client side JavaScript library which could be used for developing custom blocks and code generators as per the application requirements.

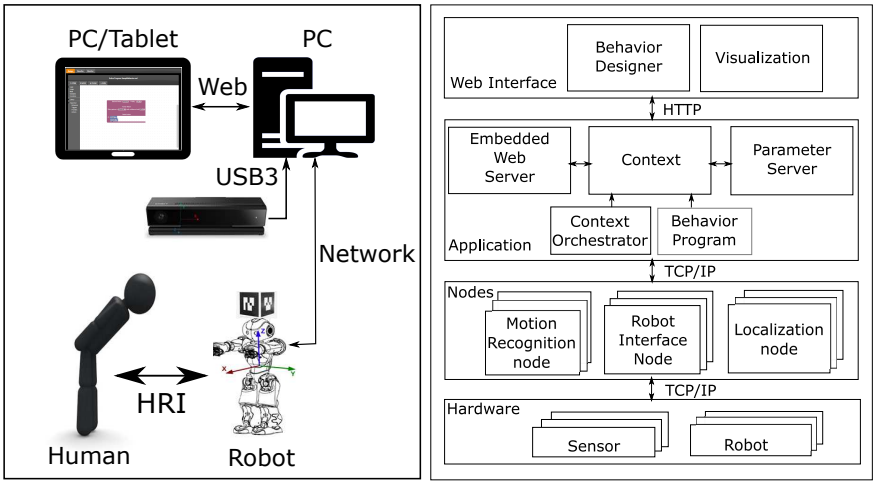
3 Motion Driven Behavior Interface

We propose a light weight interface for designing human motion driven behaviors taking inspiration from distributed architecture [17] and intuitive visual programming techniques [8].

3.1 System Architecture

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

- **Application Components**
  - *Context*: The application context contains the complete description of the world. It contains latest information about all the robots including their location, sensor data, status etc., It also contains information



(a) System Setup (b) Architecture

Fig. 1: Indriya behavior interface

about all the humans in the environment along with their active motions/gestures as supplied by the motion recognition modules.

- *Parameter Server*: The parameter server acts as a central repository for managing the parameters of the system and of the distributed components.
- *Embedded Web Server*: The web server embedded in the application serves the file and data requests from the web client.
- *Context Orchestrator*: The orchestrator keeps the Context uptodate by synchronizing with information of the robots and humans published by the distributed components.
- *Behavior Program* : A dynamic component that will be created when the user starts the program he/she designed using the user interface. The declarative description of the behavior described in Section 3.2 is parsed in order to create a memory model. The behavior program node monitors the application context for the motion triggers and invokes the corresponding robot actions according to the way it is being described in the program.

– **Distributed Components** : These are nodes in the system each with a specific goal that can be started/stopped at any time during the entire application life-cycle without affecting the other nodes or the system. All the nodes will communicate with the application using message passing techniques. They can run in any machine inside the network.

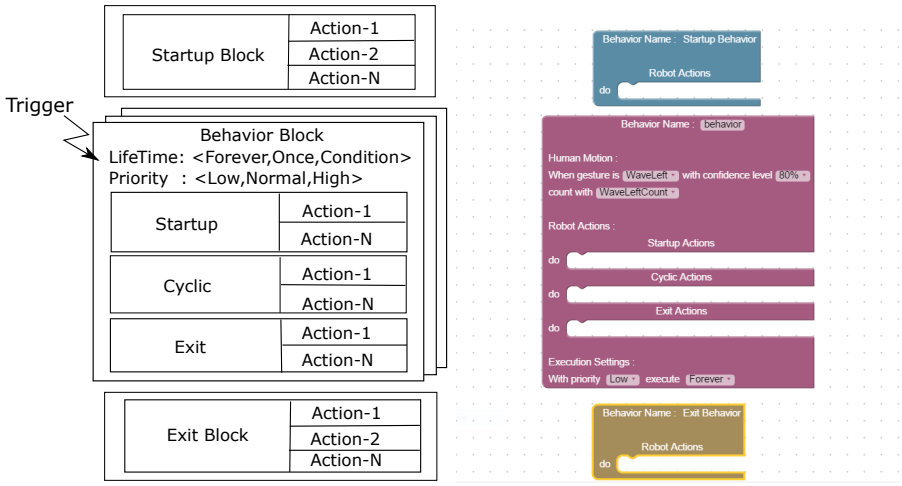
- *Motion Recognition Node* : A dedicated node that interacts with a motion recognition sensor and sends the detected motions and gestures to the application. Additionally each motion recognition module registers a set

of motions/gestures that could be detected with the sensor associated with it.
    - *Robot Interface Node* : A dedicated 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* : A dedicated node which uses the perception system to compute the position and orientation of the robot and humans in the environment.
- **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.
    - *Behavior Designer*: The Behavior designer surface 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 designed behavior will be encoded into a declarative XML format and sent to the server when the user request to start the program. The designer offers a full range of capabilities like Create/Edit/Delete/Save behavior programs.
    - *Visualization*: The visualization could be used to see the interaction of the human and robot inside a virtual 3D environment.

3.2 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:

- *Start-up and Exit Blocks*: The start-up block will be executed once when the user starts the program. The user can add a set of actions to be performed when the program 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 start-up and exit blocks in a program.
- *Behavior Block*: The behavior block is composed of
    - A **trigger** that activates this block. The trigger source could be either of human presence/absence, gesture, vicinity of human, verbal command etc.,
    - The **lifetime** of each behavior block could be configured to run only once, forever or until a condition is met.
    - The **priority** of the block could be set to low, normal or high and the execution is done based on fixed-priority pre-emptive scheduling.



(a) Conceptual Model (b) Block Implementation

Fig. 2: Behavior program structure

- Similar to the behavior program level, at each behavior block level a set of **startup** and **exit** actions could be set which would be executed only once during the creation and termination respectively. The **cyclic** actions will be performed each time the trigger condition is met.

3.3 Experimental Setup

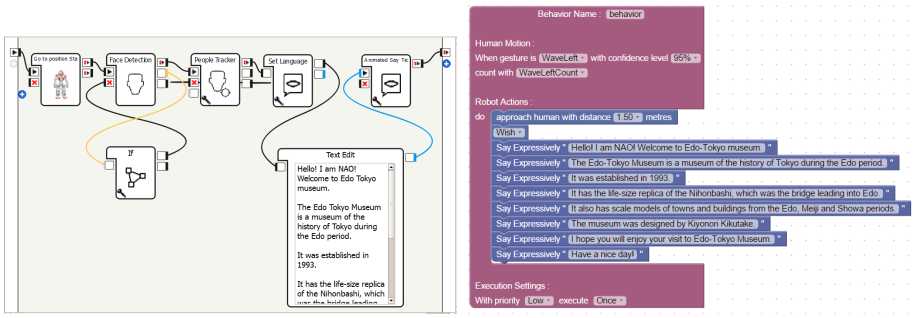
The entire framework has been implemented using open network communication standards powered by ZeroMQ [11] and message serialization using the Google's protocol buffers [9]. The Kinect for Windows V2 is used as the motion recognition system. A set of gestures are created using the visual gesture builder that comes with Kinect SDK [1]. The process involves capturing the motion clips, tagging the clips with the appropriate gestures and training the gesture recognizer. The trained gestures are then exported as a visual gesture database and integrated with the motion recognition node. The humanoid robot NAO H25(V50) powered by the latest NaoQi OS V2.1.3 is used for evaluating the interaction scenarios. A set of actions of the NAO humanoid robot are developed as python scripts. The localization of the humanoid robot is performed by combining the marker detection using augmented reality toolkit ALVAR [15] and a simple 2-DOF kinematic model of the robot from torso to the head. 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 code of the software is available as open source at https://github.com/praveenv4k/ICSORO-2015

(a) NAO as museum guide (b) NAO as therapy facilitator

Fig. 3: Experiment Setup

3.4 Example Scenarios

In this section we describe two scenarios to demonstrate how our behavior design system could be used for realistic cases which would be otherwise extremely difficult to realize using existing methods for a novice programmer.



(a) Using Choregraphe (b) Using our behavior interface

Fig. 4: NAO as museum guide

NAO as museum guide: 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. We would like to use this scenario to compare the expressiveness and intuitiveness of behavior description with Choregraphe [3] shown in Fig 4a and with our interface shown in Fig 4b.

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 4a, at first the robot keeps looking for people in its vicinity at the cost of its

power. Once it detects a person, it stops looking for people and start approaching (tracking) the person until a fixed distance with the person is reached. After this the tracking block has to be stopped and now the robot will actually start explaining the history of the museum.

Using our behavior interface, the definition of this scenario is straightforward as shown in Fig 4b. The framework 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 or a configured gesture trigger arrives, the behavior program retrieves the dynamic position of the robot and of the human. Using this information, the robot is driven 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 and he/she can just focus on the scenario rather than thinking about the details of the data flow. The experiment setup for this scenario is shown in Fig 3a.

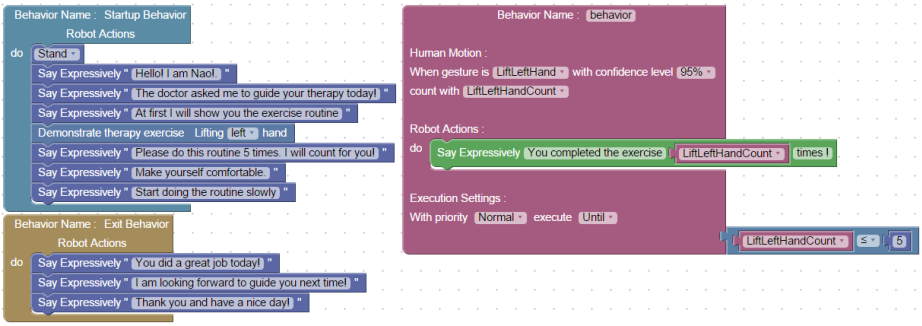


Fig. 5: NAO as therapy facilitator

NAO as therapy facilitator: 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.

This scenario cannot be realized using Choregraphe since it does not explicitly take into account the motion recognition. A reference implementation of such a scenario using our behavior interface is shown in Fig 5. In the startup behavior an introduction about the exercise routine and a demonstration of the same is performed by the robot. Then each time the patient performs the exercise, the robot notifies the progress of the exercise by announcing how many times the patient has completed the exercise. Once the exercise routine is completed (say lifting left hand 5 times), the robot gives some closing comments about

the routine. The experiment setup for this scenario is shown in Fig 3b. The experiment result of the scenarios are uploaded to http://youtu.be/gdbR199ejrg

4 Conclusion

The abundance of smart devices and sensors in the smart home and public environments provide rich information about the human motions. This information could be used for an immersive and personalized human robot interaction experience. The people from interdisciplinary fields wanting to develop a rich interaction scenario find it difficult to use the existing technology as it requires strong background in programming. Any new programming paradigm designed for such purposes should find a correct balance between simplicity and expressiveness which was the main motivation behind our proposal. We believe that our proposal could be used for studying various aspects of HRI such as efficiency of the system, cooperativeness, social acceptance etc., There exist some open questions and challenges to be addressed however. The preliminary challenges are to find a set of all possible human motions that could be understood from the sensors distributed around in the environment and to identify all possible actions a robot could perform to interact with human in a social interaction scenario. The next question is to find the spectrum of interaction scenarios this kind of programming interface could cover. It is also extremely important to evaluate the usability of the system by making statistical analysis collected on a set of participants and naive users.

5 Prospective work

This is a work in progress and for the moment we have evaluated our system for the Kinect motion capture system working seamlessly with the NAO humanoid robot for a set of predefined gestures and robot actions. We are planning to develop an extensive database containing commonly encountered gestures and also an extensive set of primitive robot actions. Additionally we are also planning to integrate our system to work with other modes of motion recognition like inertial measurement units (IMU), accelerometers and gyroscopes that are available in smart-phones and wearable devices. Similarly we are planning to integrate our system with other robots like Turtlebot and Pepper which we expect to receive soon.

6 Acknowledgments

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. This work has been supported by Student Exchange Support Program of Japan Student Services Organization (JASSO).

References

1. Kinect for windows. https://www.microsoft.com/en-us/kinectforwindows/. Accessed: 2014-11-20.
2. Aldebaran. https://www.aldebaran.com/en. Accessed: 2014-11-15.
3. Aldebaran. Nao humanoid robot. https://www.aldebaran.com/en/humanoid-robot/nao-robot-working. Accessed: 2014-11-15.
4. Vincent Berenz and Kenji Suzuki. Targets-drives-means: A declarative approach to dynamic behavior specification with higher usability. *Robotics and Autonomous Systems*, 62(4):545–555, 2014.
5. Enric Cervera, Amine Abou Moughlbay, and Philippe 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.
6. Kerstin Dautenhahn. Methodology and themes of human-robot interaction: a growing research field. *Int. J. of Advanced Robotic Systems*, 4(1):103–108, 2007.
7. Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275, 2007.
8. Google. Blockly. https://developers.google.com/blockly/. Accessed: 2015-04-31.
9. Google. Protocol buffers. https://developers.google.com/protocol-buffers/. Accessed: 2015-03-05.
10. Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *Cybernetics, IEEE Transactions on*, 43(5):1318–1334, 2013.
11. iMatix. Zeromq. http://zeromq.org/. Accessed: 2015-02-25.
12. MIT Media Lab. Scratch. https://scratch.mit.edu/. Accessed: 2015-05-15.
13. David Maier, Anja Hornung, and Maren Bennewitz. Real-time navigation in 3d environments based on depth camera data. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS Int. Conf. on*, pages 692–697. IEEE, 2012.
14. Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.
15. VTT Technical Research Centre of Finland. Alvar : A library for virtual and augmented reality. www.vtt.fi/multimedia/alvar.html. Accessed: 2015-02-20.
16. Ronald Poppe. Vision-based human motion analysis: An overview. *Computer vision and image understanding*, 108(1):4–18, 2007.
17. Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. *ICRA workshop on open source software*, 3(3.2):5, 2009.
18. Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE Int. Conf. on*, pages 1–4. IEEE, 2011.
19. Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Matthew Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake. Efficient human pose estimation from single depth images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(12):2821–2840, 2013.
20. Reid Simmons and David Apfelbaum. A task description language for robot control. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ Int. Conf. on*, volume 3, pages 1931–1937. IEEE, 1998.

21. Haibin Yan, Marcelo H Ang Jr, and Aun Neow Poo. A survey on perception methods for human–robot interaction in social robots. *Int. J. of Social Robotics*, 6(1):85–119, 2014.