

An intuitive interface for designing social behaviors based on human motions

Praveenkumar Vasudevan¹ and Gentiane Venture²

¹ Graduate Student, École Centrale de Nantes, Nantes, France,
`praveenv4k@gmail.com`

² Associate Professor, 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 architecture 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

1 Introduction

The richness and diversity of Human Robot Interaction (HRI) has been described in [10] 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 [11] in his extensive survey proposed two main types of HRI namely remote interaction or teleoperation and proximate interaction. The latter is particularly important where the humans and the robots are co-located (for example, service robots may be in the same room as humans). 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. In his survey on social robotics, Yan et al. [25] 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 shown as entertainers [3], educators [3], caring agents [1] and personal assistants [5][4]. 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 requirements. Hence consideration of augmenting exteroceptive sensors that are commonly available in the smart home/public environments is important.

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 to develop an application independent experimental platform 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 using commercially available Kinect [16] RGB-D cameras for the relative localization of human and robot and also as a gesture recognition system.

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 three decades has been presented in the surveys [18][20]. 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 [16] team which are capable of accurately predicting the 3D positions of body joints using single depth images without using any temporal information [23]. Understanding of human motion is not complete if the action of the human could not be inferred. In the survey by Microsoft research team [14], a background study on various algorithms used for human activity analysis is presented. Recently data-driven machine learning approaches like neural networks, support vector machines, clustering, decision trees

and Bayesian networks have proven to be successful with recognition accuracy as high as 94.9% [16].

The localization of humanoid robots is a challenging issue, due to rough odometry estimation, noisy onboard sensing, and the swaying motion caused by walking [9]. The Point cloud library [22] which is one of the most widely used 3D perception software library, implements ready to use probabilistic tracking algorithms. Studies on robot localization, obstacle mapping, and path planning in multilevel 3D environments by equipping Nao with a consumer-level depth camera have been reported in [17]. Localization and motion planning in smart home environment using an external Kinect sensor have been proposed in [9]. 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 [19] 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 [2] which allow non-programmers to create robot applications using a set of pre-built behavioral blocks. These programs are very intuitive and allow the users to realize complex sequence of movements and sequential behaviors. However 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 [24], scripting techniques like Universal Robotic Body Interface (URBI) [7] and middlewares ROS [21] have been proposed in the literature. Though these programs provide modular and distributed architecture, support multiple sensors and robots etc., all these tools require high level of skill in robotics and programming. Recently non-domain-specific solution like Target-Drives-Means is proposed in [8], 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 [6][12]. 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 [12] 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 [21] and intuitive visual programming techniques [12].

3.1 System Architecture

The important components of the system are shown in Fig 1

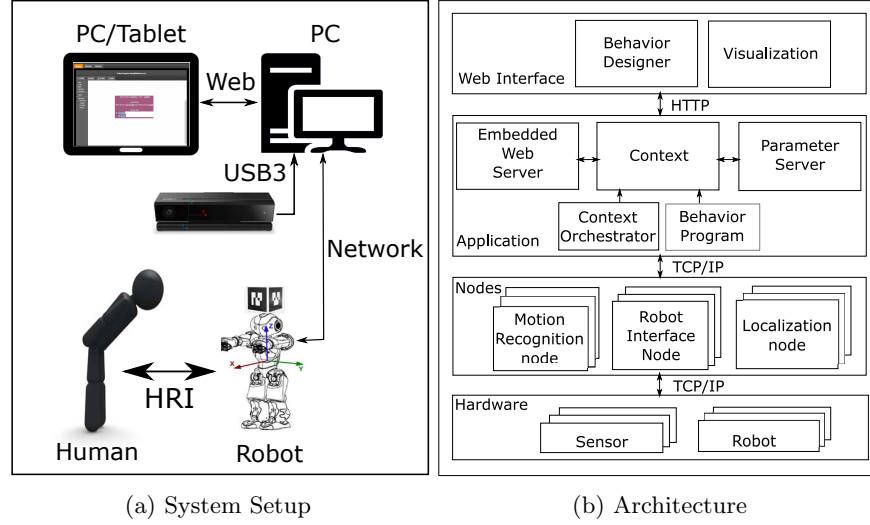


Fig. 1: System Architecture

– 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 and status. It also contains information 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 collect upto date information about the robots and humans in the environment from the perception system and updates the Context.
- *Behavior Program Node* : 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 gestures and motions to the application. Additionally each motion recognition module registers a set of actions/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 actions that could be invoked on the robot associated with it.
 - *Localization Node* : A dedicated node which uses the perception system to resolve and publish the current position of the robot and the human which is very important for interaction.
- **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 using the set of motion capabilities registered by the active motion recognition nodes and the set of robot action capabilities registered by the active robot interface nodes. The behavior designed using the designer 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 behavior program whose conceptual model shown in Fig. 2a is composed of:

- *Startup Block*: 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. This block is optional and there cannot be more than one block of this kind 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 etc.,
 - The **lifetime** of each behavior block could be configured to run only once, forever or until a condition is met.

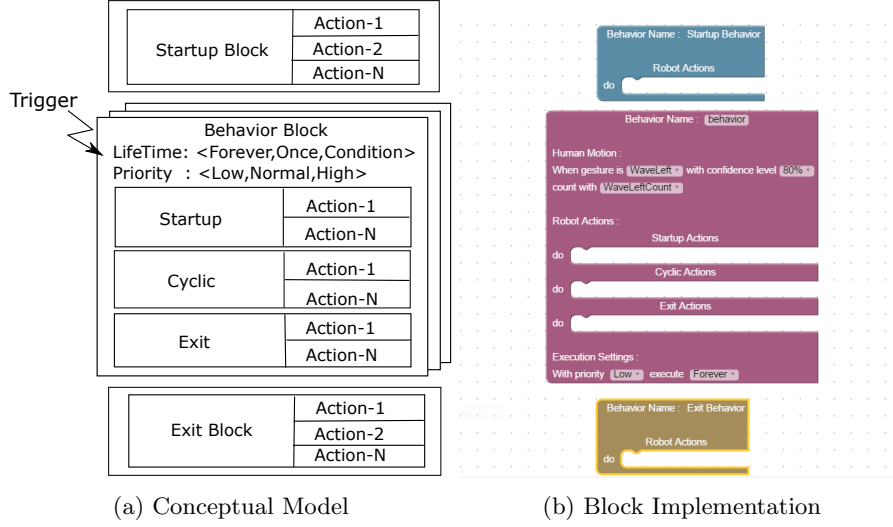


Fig. 2: Program Structure

- 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. If a higher priority behavior request to use a resource (i.e. robot) which is being used by a lower priority behavior, the higher priority preempts the lower priority behavior.
- 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.
- *Exit Block*: The exit block similar to start up block will be executed once when the lifetime of all the configured behavior blocks expire. This block is optional as well and there could be only block of this kind in a program.

The block level implementation of the conceptual model is shown in Fig. 2b

3.3 Experimentatal Setup

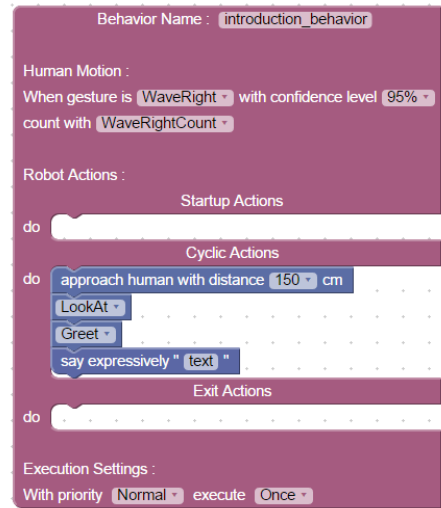
The entire framework has been implemented using open network communication standards powered by ZeroMQ [15] and message serialization is done using the Google Protocol Buffers [13] which is a language-neutral, platform-neutral, extensible mechanism for serializing structured data. The Kinect for Windows is used as the motion capture system. A set of gestures are created using the Visual Gesture Builder that comes with the Kinect Sdk [16]. The process involves capturing the clips of the motions, 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 powered by the latest NaoQi OS V2.1.3 is used for evaluating the interaction scenarios created using behavior design interface. A set of actions of the Nao humanoid robot are developed as python scripts that are being named corresponding to the actions. The localization of the humanoid robot is performed by combining the marker detection using Augmented reality toolkit ALVAR [19] and a simple 2-DOF kinematic model of the robot from the 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 operating system. The source code of the software is available as open source at <https://github.com/praveenv4k/ICSORO-2015>

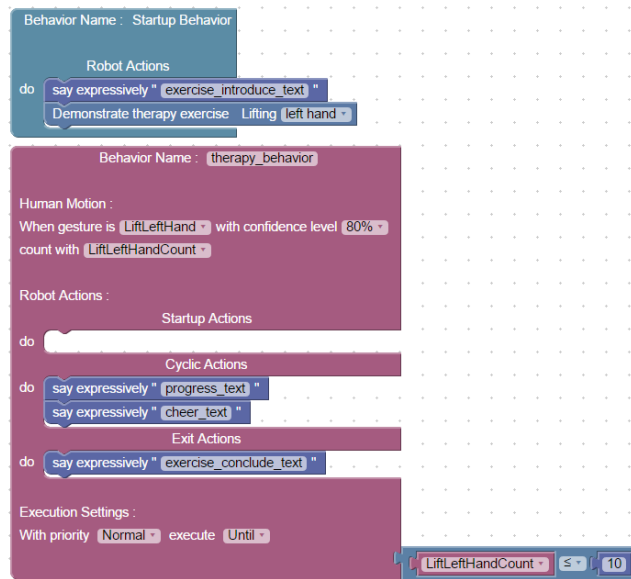
3.4 Example Scenarios

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

Scenario 1: NAO as a demonstrator The NAO humanoid robot is a demonstrator 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 [2] shown in Fig 3a and with our interface shown in Fig 3a. The first striking difference is Choregraphe uses a familiar flow chart based programming model and it has a huge library of primitive blocks to build complex motion patterns. Our interface instead uses the intuitive visual programming paradigm traditionally used for learning programming. The dataflow between the various components in the flow chart based systems gives lot of flexibility however not so easy for a beginner to start with. The data flow for this scenario is not straight forward when designed using choregraphe. As could be noticed from the figure, at first the Robot first keeps looking for people in its vicinity at the cost of its power. Once it it detects it stops looking for people and start tracking the people by keeping a fixed distance with the person. Once the target is reached, this block has to be stopped and now the robot actually start explaining the history of the museum. Using our behavior interface, the definition of this scenario is straight forward. The framework equipped with kinect sensor takes care of the detection of people and giving the relative localization. Once the person is detected, the robot is driven towards the person. After coming into the proximity of the person, the robot starts explaining the history of museum. Such a scenario could be easily designed using our behavior interface as shown in Fig 3a. The behavior block is configured to execute only once. The set of actions to be performed on observing the gesture trigger includes approaching the human keeping a distance of 1.5 m from him/her. Then look at the human, greet first and say expressively the history of the museum. It could be noticed that it is straight forward and intuitive.



(a) NAO as demonstrator



(b) NAO as therapy facilitator

Fig. 3: Example Scenarios

Scenario 2: NAO as a 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 includes the robot to give an introduction about the exercise, then to interactively report the progress of the exercise and to notify the completion. A reference implementation of such a scenario using our behavior interface is shown in Fig 3b. In the startup behavior an introduction about the exercise routine and a demonstration of the same will be performed by the robot. Then each time the patient performs the exercise, the robot notifies the progress of the exercise. Once the exercise routine is completed (say lifting left hand 10 times), the robot gives some closing comments about the routine.

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 only 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. These will be made available to the end user through the intuitive programming interface which could be then used for defining complex motion driven behaviors. 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 would like to integrate other system with other robots like Pepper and Turtlebot 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 to realize this work and participating in the experimentation. This work has been supported by Student Exchange Support Program of Japan Student Services Organization JASSO.

References

1. Aldebaran, ASKNao: Autism solution for kids Nao, <https://asknao.aldebaran.com>, Accessed: 2014-12-12.
2. Aldebaran, Choregraphe software, <http://www.aldebaranrobotics.com/en/Discover-NAO/Software/choregraphe.html>, Accessed: 2014-11-20.
3. Aldebaran, Nao humanoid robot, <https://www.aldebaran.com/en/humanoid-robot/nao-robot>, Accessed: 2014-11-15.
4. Aldebaran, Pepper humanoid robot, <https://www.aldebaran.com/en/a-robots/who-is-pepper>, Accessed: 2015-05-02.
5. Aldebaran, Project romeo, <http://projetromeo.com/en/scenario>, Accessed: 2015-04-20.
6. Lifelong Kindergarten Group at the MIT Media Lab, Scratch, <https://scratch.mit.edu/>, Accessed: 2015-05-15.
7. J. C. Baillie: Urbi 2: Introduction to concurrent real-time programming. In *Proceedings of the Third International Workshop on Software Development and Integration in Robotics*. IEEE ICRA, May 2008.
8. 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.
9. Enric Cervera, Amine Abou Moughlbay, and Philippe Martinet: Localization and navigation of an assistive humanoid robot in a smart environment. In *IEEE International Workshop on Assistance and Service Robotics in a Human Environment*, 2012.
10. Kerstin Dautenhahn: Methodology and themes of human-robot interaction: A growing research field. *International Journal of Advanced Robotic Systems*, Vedran Kordic, Aleksandar Lazinica and Munir Merdan (Ed.), 2007.
11. Michael A. Goodrich and Alan C. Schultz: Human-robot interaction: A survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.
12. Google Blockly, <https://developers.google.com/blockly/>, Accessed: 2015-04-31.
13. Google Protocol buffers, <http://zeromq.org/>, Accessed: 2015-03-05.
14. J. Han, L. Shao, D. Xu, and J. Shotton: Enhanced computer vision with Microsoft Kinect sensor: A review. *IEEE Trans. Cybernetics*, 43(5), October 2013.

15. ZeroMQ, <http://zeromq.org/>, Accessed: 2015-02-25.
16. Microsoft Kinect for Windows, <http://www.microsoft.com/en-us/kinectforwindows/>, Accessed: 2014-11-20.
17. Daniel Maier, Armin Hornung, and Maren Bennewitz: Real-time navigation in 3d environments based on depth camera data, In *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, Osaka, Japan, November 29 - Dec. 1, 2012, pages 692–697, 2012.
18. Thomas B. Moeslund, Adrian Hilton, and Volker Krger: A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(23):90 – 126, 2006. Special Issue on Modeling People: Vision-based understanding of a persons shape, appearance, movement and behaviour.
19. ALVAR - A library for virtual and augmented reality, www.vtt.fi/multimedia/alvar.html, Accessed: 2015-02-20.
20. Ronald Poppe: Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(12):4 – 18, 2007. Special Issue on Vision for Human-Computer Interaction.
21. 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.
22. Radu Bogdan Rusu and Steve Cousins: 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, 2011.
23. Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake: Efficient human pose estimation from single depth images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2821–2840, 2013.
24. Reid G. Simmons and David Apfelbaum: A task description language for robot control. In *Proceedings 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications, October 13-17, 1998, Victoria, BC, Canada*, pages 1931–1937, 1998.
25. Haibin Yan, Jr. Ang, Marcelo H., and Aun Neow Poo: A survey on perception methods for humanrobot interaction in social robots. *International Journal of Social Robotics*, 6(1):85–119, 2014.