Lehigh Instrument for Learning Interaction (LILI): An Interactive Robot to Aid Development of Social Skills for Autistic Children

Mooi Choo Chuah\textsuperscript{1}, Daniel Coombe\textsuperscript{1}, Christopher Garman\textsuperscript{1}, Cassandra Guerrero\textsuperscript{2}, John Spletzer\textsuperscript{1}

\textsuperscript{1}Department of Computer Science and Engineering; Lehigh University; Bethlehem, PA 18015
\textsuperscript{2}Department of Computer Science and Information Technology; Kutztown University; Kutztown, PA 19530

Abstract—Recent studies show that autistic children tend to speak and interact more in the presence of an interactive robot. Unfortunately, most of the robotic experiments were conducted in highly controlled clinical settings or limited selected home environments due to the high deployment cost. In this paper, we design a low-cost interactive robot that can be readily deployed to home environments. Our robot, called Lehigh Instrument for Learning Interaction (LILI), interacts with users via gestures, voice commands, and an animated speaking avatar. LILI can recognize users’ faces, and her motion can be controlled either via gesture or voice. We describe both the system and software architectures of LILI. Experimental results are also presented.

Keywords: Human-robot interaction; interactive robot; autism spectrum disorder.

I. INTRODUCTION

According to the Centers for Disease Control and Prevention, 1 out of 68 children born in US suffer from autism spectrum disorder (ASD) [1]. Often, such children suffer from certain developmental deficits that include poor social interaction and conversational skills [2-3]. Common conversational skills of children affected by ASD include difficulty managing turn-taking and discourse topics, using inappropriate speech style, and difficulty inferring what information is relevant or interesting to others. They also often lack perception of non-verbal attention cues, affective expressions, eye contact and expressive emotions. Such deficits impact their abilities to function independently in social, occupational and other important areas of their lives. Early intervention is considered critical for such children [4-5].

Various human-delivered intervention strategies have been proposed in the past e.g. [3,4]. These strategies vary in terms of (a) specific targeted behaviors, (b) whether such behaviors are spontaneously initiated by a child or explicitly elicited via highly structured repetitive instruction, and (c) the settings in which training or reinforcement takes place, e.g. in highly controlled clinical settings or more natural environments such as in a child’s home. Intervention often rewards children for performing the targeted behavior, e.g. by allowing a child to engage in a preferred activity or play with a preferred toy.

Research has shown that the use of child-preferred reinforcers often leads to greater social interaction [6,7]. Preliminary research has also shown that children with ASD tend to interact easily with robots [8-10]. Social robots are designed to evoke social behaviors and perceptions in the people with whom they interact. The study in [7] shows that embedding social interaction within the delivery of preferred reinforcers increases production of target behaviors while the study in [10] shows that children with ASD spoke more in general and directed more speech to the adult when the interaction partner was a robot compared to a human or computer game interaction partner. Children spoke as much to the robot as to the adult interaction partner. In another study [13], they used a co-robotic intervention platform and environment specifically developed to accelerate developments in early joint attention skills [11-12]. Their study found that children with ASD demonstrated an attentional bias toward the robot as opposed to a human therapist and that a closed-loop autonomous robot mediated joint-attention intervention system could be dynamically adapted based on the performance of a child.

Figure 1: LILI with and without the outer shell, with primary hardware components highlighted.

Even though all these studies show promising trends in using robots to improve the learning speed of various functional skills for ASD children, e.g. joint attentions, social interactions, and conversational skills, the cost of deployment in typical classrooms or in homes is relatively expensive. Thus, a low cost robot platform that can be used to train ASD children’s or teenagers’ social cues and conversational skills is
highly desirable. Such motivation drives the design of our own robot, called Lehigh Instrument for Learning Interaction (LILI), which uses face, gesture, and voice recognition to interact with users. She speaks through an animated talking avatar displayed on a monitor and is also capable of moving through her environment. In the following sections, we will discuss LILI’s software architecture, hardware, and describe how each component works.

II. RELATED WORK

There have been numerous research and educational robots that have been developed based upon the iRobot Create chassis. Indeed, in our previous research we developed the Automated Asset Locating System (AALS) based upon the Create platform for use in inventory management [24]. However, the industry standard is the TurtleBot platform [25] developed and commercialized by Willow Garage, and currently supported by the Open Source Robotics Foundation. This is a turnkey platform that integrates a Microsoft Kinect sensor and the Robot Operating System (ROS) as its development environment [26]. ROS provides efficient mechanisms for inter-process communication, high fidelity visualization, and a tremendous repository of add-on packages. However, it has a relatively steep learning curve and significant code footprint. As a result, we chose the route of custom system development. This was driven by a desired form factor, computational requirements, and the desire for a flexible, higher level software development environment.

Other development platforms were considered, but disqualified from consideration based upon cost. For example, Herb 2.0 is a bimanual mobile manipulator developed at the Personal Robotics Labs at CMU [17]. It can learn, adapt and exploit its environments to perform tasks in the kitchen and office environments. However, its high cost ($50K) means it cannot be widely deployed in typical homes. Nao is a humanoid robot designed by Aldebaran Robotics in France. This humanoid robot had been designed purposely to look approachable and portray emotions like a toddler. Researchers have used Nao to train children with ASD on how to understand and perceive human emotions, a skill which such children typically lack. In [18], a researcher developed a robot-assisted protocol which yields more objective and effective autism diagnosis using a Nao robot. The Nao robot is programmed to call a child’s name such that observer can then evaluate a child’s ability to transfer attention to the source of the call. Additional tasks performed by the Nao robot included performing a task that a child can imitate as well as calling that child’s attention to an object etc. Other researchers in Malaysia have also developed their own protocol to evaluate the initial response of autistic children in human-robot interaction therapy with a Nao robot [19]. Despite these positives, the $8K price tag for Nao is still excessive to most families.

Lastly, researchers from USC have designed their own humanoid robots and used them to augment interventions for children with ASDs [20,21]. They designed the Behavior-Based Behavior Intervention Architecture where a robot is used to observe the behavior of a child through a collection of on-board sensors or wearable sensors and then respond accordingly based on their sensed behaviors. However, their designed robots are not yet on the market. In contrast to these commercial systems, the bill of materials for LILI is less than $1,000, including onboard computing. This was the final driver in our system development.

III. LILI SYSTEM DESCRIPTION

The primary objective of our work is to develop a low-cost robot capable of interacting with ASD affected children/teenagers via face, voice, and gesture recognition. In this section, we first give an overview of the LILI system architecture with its primary system components. Then, we will describe in more detail the various software modules developed to provide the five main features, namely (i) face recognition, (ii) gesture recognition, (iii) voice recognition, (iv) talking avatar, and (v) robot control.

A. System Architecture

LILI consists of an iRobot Create base for mobility [14], an Asus Xtion PRO sensor for perception [22], and a video monitor for avatar interaction. All computation is handled by the onboard computer which integrates an i3-3220T 64-bit processor running Windows 7 Enterprise. The system architecture is shown in Figure 2.

![Figure 2: System Architecture diagram showing the interactions between hardware and software components.](image-url)
Master Planner module is ultimately responsible for system control. It directly controls the movement of the iRobot Create, and communicates with the other hardware using sub-processes `VoiceMonitor.py`, `KinectMonitor.py`, and `phrasesToSay.py`. The voice recognition module (shown in blue) uses the Microsoft Speech Platform to analyze the speech of a user and determine if a valid phrase had been said. If a phrase was said, that information is passed onto the subprocess that monitors speech commands. If the command came in within three seconds of the word "Lily" then the corresponding action is sent on to the Master Planner to be executed. The Kinect Monitor module (red) uses openCV, openNI, and Nite to interpret information from both the depth sensor and the RGB camera. The RGB camera information is directly used for face recognition and detection. Then, once a decision has been made about the user, the user's identity is passed on to the Master Planner. The depth sensor information is passed through the Nitepy programs to be converted to coordinate points that can be used to determine what a user's gesture. The gesture is then interpreted, and the corresponding action is sent to the Master Planner to be executed. The PhrasesToSay module (green) will verify if the keyword sent by the MasterPlanner is valid and instruct the BaldiSync program to say the appropriate speech and animate the face appropriately.

Gesture recognition is accomplished using depth data from the Xtion PRO. A gesture is defined by a series of states in a finite state machine which a user must follow for her/his gesture to be recognized. When any defined gesture is recognized by the Kinect Monitor (KM), a command is issued to the Master Planner, where it is placed on a queue to be executed. Currently, our supported gestures include a left or right wave, a follow and stop follow, and quit.

**Figure 3: Software Architecture. Each sub-process is color coded.**

**B. Gesture Recognition**

For perception, LILI employs an Asus Xtion PRO Live sensor. The Xtion PRO consists of a depth sensor, an RGB camera, and two microphones in a configuration mirroring Microsoft’s Kinect. For gesture recognition, point cloud data are used to track skeletons in conjunction with OpenNI [27] and NITE [28]. We use the OpenNI interface to acquire both color and depth images from the Xtion PRO. NITE is a middleware for OpenNI that processes only the depth data from the sensor and uses these to find and track the user’s approximated skeletons. An example of skeleton tracking is shown in Figure 4.

**Figure 4: An example of skeleton tracking overlaid on the point cloud data**

**C. Face Recognition**

RGB color images from the Xtion PRO are processed by OpenCV to identify users via face recognition. This system works in two stages: 1) detect the presence of faces, and 2) attempt to identify any detected faces.

The face detector uses Haar cascades to identify the location of faces within the image. For both robustness and computational efficiency, we leverage the depth sensor and limit the search to a region of interest around the heads identified by the skeleton tracker. An example of the face detection can be seen in Figure 5. We identify faces using the FisherFace algorithm from OpenCV’s Face Recognizer API. The recognizer must first be trained with a variety of images for each face. These images must all be in grayscale, have the same dimensions, and are best when taken from the front and in similar lighting across the different users. This helps to eliminate lighting bias during identification. The image being tested against the database must also be in grayscale and have the same dimensions as the training images. Our face recognizer uses an image size of 240x240 pixels. An example of the FisherFaces used in our database is shown in Figure 6.
Figure 5: Example of a face being detected inside the region of interest.

Figure 6: An example of FisherFaces used for training the database and for testing faces.

After processing an image, the recognizer returns a guess and a confidence value. We determined through testing that a confidence level of 2000 provided reliable recognition, and associated the identity to the respective skeleton. Note that this threshold is highly camera dependent. If the identification is not successful, the recognizer makes additional attempts once per second, up to 10 times to identify a single user. The limit of 10 attempts is again based upon empirical testing to maximize the chance of true positives while keeping the potential for false positives to an acceptable level. The recognizer will continue to attempt until either all users have been recognized or no attempts remain.

D. Speech Recognition

Speech recognition is implemented in C# through the Microsoft Speech Platform SDK 11 and takes input from the microphones on the Xtion PRO. The interface for managing the speech recognition is packed into an assembly (a C# DLL) called LILIVoiceCommand.dll. The tools and the voice recognition inside this DLL are made available to python using Python for .NET and are used in the file VoiceMonitor.py (VM). This DLL allows LILI to recognize her own name and the following commands: “Move right”, “Move left”, “Follow me”, “Stop”, and “Goodbye”.

The VM sends recognized commands to the Master Planner, which places the commands in a queue. The Master Planner will attend to whatever command is first in the queue, whether it arrives via speech or gesture recognition.

E. Animated Talking Avatar

LILI speaks using BaldiSync [15], an end-user program which converts text to speech using Festival and lip-syncs an animated face to the text. An example of the animated face can be seen in Figure 7. To circumvent the “end-user” properties of the program, we modified the source code (bsync.tcl) so that the text-to-speech conversion can be executed using only the Return key and not a series of mouse clicks. We also wrote a C++ program that simulates keyboard input into BaldiSync’s text window, including the necessary tags before and after the text to select the desired voice. The modification combined with the simulated keystrokes allows BaldiSync to be run as a single sub-process to the Master Planner without the need of a user to enter in text and click the buttons to turn it into animation.

Figure 7: LILI’s face during speech.

PhrasesToSay.py contains built-in phrases that can be converted to speech and animation. We reference these phrases with specific keywords. We also have the users’ names incorporated into our database. This allows us to integrate a user’s name into the built-in phrases to personalize user experience. When the Master Planner is going to execute a command, it issues a keyword to PhrasesToSay.py so that LILI will speak the associated phrase.

F. Robot Control

LILI moves using an iRobot Create with a differential drive system as its base. Our interface to the Create is also through Python. LILI’s velocity is constrained by the iRobot Create to 0.5 m/s. When the robot is ready to move, the Master Planner will check for queued commands. There are two queues: qGest and qFollow. Registered gestures and voice commands, i.e. a left wave, a right wave, or a quit command, are held in qGest, while follow and stop commands are held in qFollow. Commands from qGest are executed from
the ‘waiting’ state, while an initial follow command will change the state to ‘following’ until a stop command is received.

By default, a left or right wave will move LILI one meter to the user’s left or right. For example, when a left wave is received, the Master Planner will write to phrasesToSay.py, prompting LILI to say, “I am moving to your left.” She will make a right turn to the user’s left, drive forward one meter, turn back to the user, and wait for a new command.

When a follow command is received from the VM/KM, the KM will send the user’s coordinates to the Master Planner. LILI will attempt to follow 1.5 meters behind the user until either the user’s skeleton is lost or a stop/quit command is received. She will not respond to any other commands during the “following” state. The feedback control loop for motion planning runs at 5 Hz so that LILI can move smoothly through the environment.

To follow a user, LILI uses a sample-based planner, where sampling is done on the linear and angular velocity input spaces [16]. The predicted positions and orientations for LILI are calculated at 5 Hz for each velocity pair (v, \(\omega\)), where v ranges from 0.0 to 0.5 m/s and \(\omega\) ranges from -0.5 to 0.5 rad/s. Linear and angular velocity pairs requiring wheel velocities above 0.5 m/s are invalidated as these are beyond the limits of the Create hardware. Sample step sizes are 0.1 m/s for v and 0.1 rad/s for \(\omega\), a planning horizon of 2 seconds is used, and the distance to goal is used as the cost function for scoring the individual trajectories. Since person following is done in the robot frame, the end points of the trajectories are pre-calculated in a lookup table (LUT). This allows the desired point for \((v,\omega)\) to be quickly calculated by simply finding the minimum distance between a user’s current position in the robot frame to the points in the LUT.

Currently, no obstacle detection/avoidance behaviors are implemented on LILI. These are part of ongoing development efforts.

IV. RESULTS AND FUTURE WORK

Our current prototype implements all of the functionality outlined above. A demo of our final prototype that shows all major features being tested can be seen in [23]. Images of LILI following a user are shown in Figure 8. LILI initially queries the user for a command. The user responds by gesturing “follow”. LILI then tracks the user skeleton as outlined in Section 3.2, and follows using the controller outlined in Section 3.6 (Figures 8.b-8.c). The demonstration concludes with the user gesturing “stop follow” (Figure 8.d).

While our team is very satisfied with the progress made over the past 8 weeks, there is still significant room for future development. For example, LILI can consistently recognize gesture and voice commands. However, a limitation of the current voice recognition behavior is that LILI only attempts to recognize phrases defined in her speech recognition engine’s grammar object. As a result, voice commands are occasionally falsely recognized (e.g. ‘light’ may be recognized as ‘right’). In addition, while we are able to use the face recognition module to only accept commands from authorized users, this does not protect against extraneous voice inputs from other non-authorized users nearby. We are currently examining approaches to improve system security.

![Figure 8: Images of LiLi Following a User via Gesture Commands.](image)

The gesture database is also quite small (a total of 5 gestures). Expanding this is also a topic of ongoing development. The performance of face recognition is also highly dependent upon the ambient lighting, and we are working to improve the robustness of the face recognition algorithm.

ACKNOWLEDGMENT

C. Garman, C. Guerrero were funded by the Lehigh Smartspace NSF REU site proposal under contract number 1359280. D. Coombe is partially funded by the NSF funded SECON project, CNS 1049845.

REFERENCES


[15] CSLU Toolkit; Center for Spoken Language and Understanding; Oregon Health and Science University; Portland, OR 97239


[23] LILI’s final demo located at: http://youtu.be/PsDFz4MNGdY


[27] OpenNI available at: http://structure.io/openni

[28] NITE available from http://primesense-nite2-for-windows.software.informer.com/2/0