Using Haptic and Auditory Interaction Tools to Engage Students with Visual Impairments in Robot Programming Activities

Ayanna M. Howard, *Senior Member*, *IEEE*, Chung Hyuk Park, *Member*, *IEEE*, and Sekou Remy, *Member*, *IEEE*

Abstract—The robotics field represents the integration of multiple facets of computer science and engineering. Robotics-based activities have been shown to encourage K-12 students to consider careers in computing and have even been adopted as part of core computer-science curriculum at a number of universities. Unfortunately, for students with visual impairments, there are still inadequate opportunities made available for teaching basic computing concepts using robotics-based curriculum. This outcome is generally due to the scarcity of accessible interfaces to educational robots and the unfamiliarity of teachers with alternative (e.g., nonvisual) teaching methods. As such, in this paper, we discuss the use of alternative interface modalities to engage students with visual impairments in robotics-based programming activities. We provide an overview of the interaction system and results on a pilot study that engaged nine middle school students with visual impairments during a two-week summer camp.

Index Terms—Education, haptic I/O, functional programming, robotics.

1 Introduction

The percentage of entering college freshman reporting disabilities continues to increase in the academic environment, with a recently reported growth of 9 percent [1], [2]. Of those disabilities reported, vision impairments ranked at approximately 16 percent and among those undergraduate students with reported disabilities, only 3.9 percent majored in computer science [1]. Differences in precollege math and science education, which provides a foundation for pursuing a computing degree, are a large contributing factor to this disparity. At the precollege level, approximately 11 percent of children between the ages of 6 to 14 have a reported disability [3], [4], and yet these students took fewer science and mathematics courses than those without disabilities. These differences are generally due to the unavailability of information in accessible formats and the unfamiliarity of teachers with alternative (e.g., nonvisual) teaching methods [3]. Unfortunately, there are only a few efforts that are currently adopted to engage students with visual impairments in the computing fields at the precollege level, including the National Center for Blind Youth in Science [5], the AccessComputing Alliance [6], and Project ACE [7], which provide resources to prepare youth with disabilities to pursue higher education and computing career opportunities.

On the other hand, the field of robotics is extremely popular across generations of students. Many students with disabilities, like most students in K-12, are naturally interested in robotics [8]. Unfortunately, the scarcity of accessible interfaces to educational robots can lead to students with visual impairments not having equal participation with peers in robot-based computing activities. In fact, most interfaces that allow robot programming are built on traditional visual and keyboard-based inputs. And yet, even if you solely examine the needs of students with visual impairments, there is sufficient diversity in the interfaces required—10 percent are registered Braille readers, 27 percent are visual readers, 8 percent are auditory readers, 34 percent are nonreaders, and 22 percent are prereaders [9]. As such, our research is focused on engaging students with visual impairments by focusing on *accessible interfaces* for robot programming. Our primary strategy is to capitalize on the appeal of robotics in order to both deliver and engage precollege level students with visual impairments in computing. Our hypothesis is that as long as alternative interface technologies can be employed, a student can become an active participant in robotics-based computing activities, with the goal of encouraging them to consider future possibilities in computing.

2 Background

Over the past few years, there have been a number of efforts to ensure computers (and, more widely the Internet) are accessible to individuals with visually impairments [10]. Solutions range from software and hardware tools that utilize Braille terminals, screen magnification, screen readers, and/or synthesized speech output. Some browsers have even been developed especially for people who are visually impaired, such as BrailleSurf and BrookesTalk, which

- *A.M. Howard and C.H. Park are with the HumAnS Laboratory, Georgia Institute of Technology, TSRB 444, 85 5th Street NW, Altanta, GA 30308. E-mail: ayanna.howard@ece.gatech.edu, chungpark@gatech.edu.*
- *S. Remy is with the Computer Science and Engineering Department, University of Notre Dame, 356D Fitzpatrick Hall, Notre Dame, Indiana 46556. E-mail: sekou.remy@nd.edu.*

Manuscript received 5 Oct. 2010; revised 12 Feb. 2011; accepted 11 Sept. 2011; published online 5 Dec. 2011.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2010-10-0115.
Digital Object Identifier no. 10.1109/TLT.2011.28.

incorporate voice output to translate content from the web. Although these technologies provide support to enable students who are blind or visually impaired to access computer environments, there has not been as much support in improving the accessibility of these environments for programming activities. A significant issue is that many of the tools needed to program in graphically oriented environments are usually visual themselves. To resolve this dilemma, there have been some efforts that have focused on building support tools for graphical development environments [11], [12]. Most of these efforts use audio and speech cues to enable access to compiler and debugging tools [13], [14]. Although these mechanisms have shown to be viable resources, none of these efforts have focused on supporting the processes for learning *how* to program. So far, they still tend to focus on supporting individuals who already have some programming knowledge—primarily with the goal of engaging students at the college level.

With respect to precollege initiatives, there are few that exist. Ludi [15] organized a technology camp in which students with visual impairments were taught how to program a Lego robotic platform. In this work, students were provided with a voice-reader equipped programming environment, but students could not feel the actions of their running robot remotely—they had to follow the robot and use direct tactile sensing (i.e., touching the robot). There was therefore no means provided, beyond touch, to enable students to verify and debug program correctness beyond a successful compile. In the Chatbot program [16], students were provided a taste of computer science at the National Federation of the Blind YouthSlam by designing their own instant messaging chatbots. In this project, computer science instructors worked directly with students to go through C# programming tutorials, help them implement their ideas and debug their program. This required a low student-to-instructor ratio of approximately 4-to-1. Unfortunately, beyond these efforts, there are few existing initiatives designed to engage precollege level students with visual impairments in the computing fields. We do take guidance from Ludi's work though as her project enables students to independently learn how to program. However, we expand on this concept by providing a means for the student to use additional feedback to both enhance learning of program syntax, as well as enhance debugging of program functionality.

With respect to noncomputing-related efforts and robotics specifically, there are a number of projects that focus on using robotic platforms to provide sensory feedback to enable individuals who are visually impaired to interact with their environment. Most notable is work by Kulyukin [17], in which a mobile robot platform was used to help individuals navigate through their surroundings. In [18], Ulrich and Borenstein developed a robotic cane that can actively sense the environment and guide individuals who are blind through the environment. However, both systems were used for active contact-based guidance, and were not focused on remotely transferring dynamically sensed information to the user, which would be required if interacting with a mobile robot at any remote distance. Another concept, presented in [19], linked robotic sensing extracted in real time to transfer environmental-based haptic feedback to the user. This initial methodology showed promise in providing sensing feedback about the environment to remote users.

Unfortunately, none of the previous efforts, by themselves, focus on providing precollege age students who are visually impaired access to the full spectrum of tasks required in robotics-based programming activities. As such, we discuss our design of accessible interfaces that provide sufficient feedback mechanisms to facilitate this programming process.

3 Programming/Robot Interaction System

3.1 Learning Strategy

In traditional programming processes, we tend to utilize visual feedback to enable writing/compiling of a program, evaluation of the program output, and debugging the program based on this output. Transitioning this to the traditional robot programming processes involves expanding these steps to

1. Writing the program based on the robot command set (library),
2. Compiling the program,
3. Downloading the code onto the robot,
4. Running the code, and
5. Adapting the program based on evaluation of the robot actions.

Again, in traditional settings, these steps tend to be highly visual. Our goal is therefore to utilize additional feedback mechanisms to facilitate programming of robots for students with visual impairments. Our strategy involves partitioning the interaction space into three primary feedback components: 1) interaction/feedback during programming, 2) interaction/feedback during program execution, and 3) interaction/feedback after program execution. Based on these components, we begin by postulating three hypotheses:

- Hypothesis 1: Existing computer accessibility technology (e.g., computer screen readers and magnifiers) can be modified and integrated to provide sufficient feedback for students with visual impairments to enable the programming process.
- Hypothesis 2: Correlating haptic and/or audio feedback with real-time program execution can provide sufficient feedback to enable students with visual impairments to visualize their programmed robot sequences.
- Hypothesis 3: Enabling automated verbal feedback to summarize program output after completion can provide sufficient feedback to enable students with visual impairments to understand changes that maybe required in their program.

The following sections discuss our strategy and corresponding assessment based on utilizing the three interaction feedback components to address these hypotheses. We begin our discussion by presenting the robot programming and implementation platform used for engaging the students.

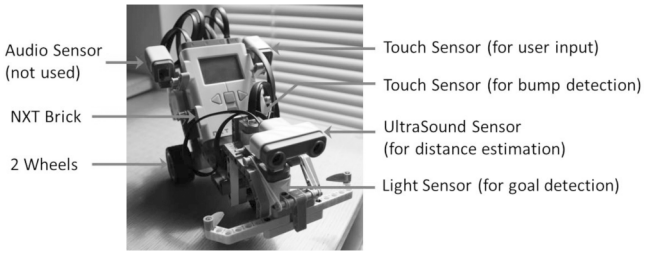

Fig. 1. Robot hardware platform for programming.

```
#include "cbvi.h"

/*
  Welcome to the task 1 programming template.
  This template has examples which should help you to talk to your robot.
*/

/*
 You may have noticed by now that there was a / followed by a * at the beginning of this section.
 This is how comments are opened.

 We will also use a * followed by a / to close a comment.

 Where ever we open a comment, we must close them.

*/

/*
Comments are used to talk to you the programmer, and sometimes remind you of things later on.
*/
```

Fig. 2. Snapshot of tutorial/programming template.

3.2 Interaction/Feedback during Programming

The first step in engaging students with visual impairments into robot programming activities is to choose a robot platform and programming environment that is accessible (or can be made accessible). As our goal is to engage students at the precollege level, we began by selecting the Lego Mindstorms NXT robot kit, which is readily available, and has a proven record of engaging students through competitions such as FIRST Lego league [20]. Although there are a number of available programming interfaces for the NXT, based on prior work by Ludi and Reichlmayr [15], we selected the Bricx Command Center (BricxCC) as a programming interface due to its accessibility attributes. BricxCC is an open-source integrated development environment (IDE) based on a text-based C-programming language called Not Quite C (NQC) [21]. The accessibility benefits of the BricxCC IDE include an environment in which to both write, compile, and debug software code all in one package, which enables use of a common set of commands and shortcut keys. Additional access for students with visual impairments includes compatibility with popular screen readers (e.g., JAWS). The BricxCC IDE also has integrated support for programming other robotic platforms. The robots used for programming in this session are prebuilt for the students to provide identical hardware platform for all participants so they can program their own software to make their robot functional. The robot is composed of one LEGO Brick computing block, two motors with wheels and built-in encoders for odometry calculation, two touch sensors to detect user input and bumping incidents, one light sensor to detect a goal on the floor, and one forward-facing ultrasound sensor to detect distance from an object (Fig. 1).

Beginning from this basic infrastructure design, we integrate the JAWS screen reader and MAGIC magnifier [22] to provide access for the students to the programming environment. The screen reader enables a direct text-to-voice conversion of the text within the programming console whereas the magnifier provides expansion of the text for those designated as legally blind[1] (i.e., a central visual acuity of 20/200 or less) but also designated as having low vision. We have also experimented with the use of open-source packages such as Orca for screen reading under the Linux operating system, which is discussed in the future works section.

Typically with introductory programming courses, tutorials are provided which enable the student to learn about programming concepts. These tutorials also introduce the student to the syntax of the particular language or languages of interest. In most cases, these tutorials are external documents and are sometimes even provided in textbook format. Based on prior work in task switching [23], we sought to minimize the cognitive switching costs associated with transitioning back-and-forth from the tutorial screen/document and programming console by embedding the tutorial directly in the comments of the actual program (Fig. 2). The tutorials were thus provided to the students as a programming template that both taught the students as well as directed them to perform various tasks. These templates were developed as a sequence of learning and evaluation opportunities that transitioned them through the learning of different programming concepts by using skills to program different robot tasks (Table 1). The tasks are designed such that each task builds on knowledge attained from the prior tasks, thus systematically increasing the skill level required to complete.

Throughout the template, we also embed question/answer assignments that require the student to provide a verbal response to the instructor. For example, when explaining the concept of comments, the template states:

```
/*
Can you tell the instructor how to make your own comments?
Raise your hand if you think you know how to make a comment
*/
```

This process is designed to gauge student progress in a nonintrusive manner, while still ensuring the student-teacher

TABLE 1
Themes of Programming Templates

	CONCEPTS	SKILL	PROGRAMMING TASK
1	Programming syntax, function of main, comments, compiling, downloading code to the robot	Creating your first program	Turn the robot on, move up once, turn the robot off
2	Sequential programming, variables, functions (robot command library)	Associating real-world straight-line movements to programming units	Turn the robot on, move up 80cm, turn the robot off
3		Associating real-world turn movements to programming units	Turn the robot on, turn left 45 degrees, turn the robot off
4		Tying learned skills together for robot control	Move forward 300cm, turn left 90 degrees, move forward 100cm
5	Debugging based on real-world inputs	Applying learned skills to an application, (e.g. correlating learning program commands to real-world robot movement)	Move the robot in the shape of a square

1. An estimated 3.5 million Americans have low vision. Out of that group, approximately one million meet the legal criteria for blindness [18].

relationship is maintained. The benefits of the embedded template approach include the possibility for students of varying skill levels to learn at their own pace within the same educational environment. Prior studies [24] have shown that this type of technology-based instruction promotes a more student centered and individualized experience and enables students to take an active role in their learning. It is anticipated that this will be beneficial in the classroom setting, especially with students having diverse backgrounds and experience levels.

During this programming phase, feedback/interaction is provided through the integration of the screen reader, programming template, and standard BricxCC environment. This type of auditory interaction enables the students to sequentially navigate through their program, with a verbal validation on their typewritten code.

3.3 Interaction/Feedback during Program Execution

In relevant work [25], it was shown that, by integrating sound and touch, access to complex mathematical graphs and tables could be provided to individuals with visual impairments. Motivated by such studies, we sought to provide similar experiences to assist students with visual impairment in programming a robotic system and in testing the designed system, by developing a process for feeling the robot's movement and the environment in which the robot is traveling. Unfortunately, a common characteristic found in conventional haptic applications is that haptic feedback is typically generated from a static source. For example, in virtual environments, elements in the world commonly use 3D models suitable for direct representation of haptic features [26]. Also, in haptic systems for real-world objects [27], the sensor system and the target object interact in a fixed environment, which confines the operational workspace of the haptic device. The primary difference between our application and other haptic-based work is that 1) the haptic forces must be generated in real time, from the robot perspective [19], and 2) the haptic device should be both mobile and portable in order to ensure individuals are free to move around with their robots and are not yoked to a fixed, stationary, location. As such, we have designed a human-robot interaction framework that utilizes multimodal feedback to transfer the environmental perception to human users who are visually impaired. Although, there are two types of haptic feedback used to provide a sense of touch—tactile feedback and force feedback,[2] we specifically target the use of vibrations for our application. As such, vibration-based haptic feedback and auditory feedback are selected as primary methods for user interaction.

The communication between the user, the robot, and the computer that decodes the signals (designated as the host PC) are connected wirelessly with Bluetooth connectivity. The detailed system architecture is shown in Fig. 3. Every time the robot receives sensor requests from the host PC, the robot transmits its sensor data (such as touch sensor, light sensor, and encoder values). Once the PC receives the sensor values, the sensor-input handler block removes noise

2. Haptic feedback is commonly represented by tactile and/or force feedback. In most haptic studies, tactile feedback is created via heat, pressure, and/or vibrations [28].

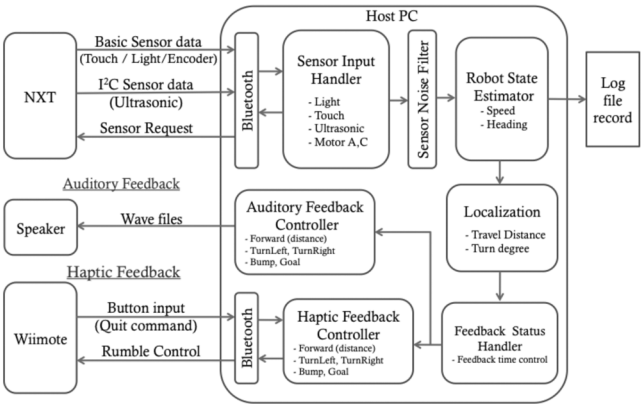

Fig. 3. System architecture and communication.

in the sensor values through the sensor noise filter block, and then the localization module transforms the values into robotic state estimates such as speed and heading using the following equations:

$$S = \begin{cases} fwd, & if\ \omega_{left} > 0, \omega_{right} > 0 \\ turnleft, & if\ \omega_{left} \leq 0, \omega_{right} > 0 \\ turnright, & if\ \omega_{left} > 0, \omega_{right} \leq 0, \\ stop, & else \end{cases} \tag{1}$$

$$\theta = \frac{1}{\text{UnitDegreeSteps}} \sum_{t=1}^{T} (\omega_{left}(t) - \omega_{right}(t)), \tag{2}$$

$$X = \frac{1}{\text{UnitDistanceSteps}} \sum_{t=1}^{T} (\omega_{left}(t) + \omega_{right}(t))/2, \tag{3}$$

where:

ω_{left} : Encoder count of left motor,
ω_{right} : Encoder count of right motor,
X : Distance of forward motion,
θ : Degree of turning motion,
S : NXT robot movement primitive.

These state estimations are then transferred to the feedback status handler module, which determines which type of haptic or auditory feedback signals to generate. These values are also recorded into a log file that is handled for providing feedback after program execution.

During program execution, both auditory and haptic feedback signals are provided to enable validation of a user's program. Based on the sensor suite integrated into the robotic platform, five feedback primitives were designed to provide both auditory and haptic feedback. For haptic feedback, a Wii remote controller (Wiimote) was used as the primary interface between the robot and the user. In the case of auditory feedback, various sounds associated with different piano notes were recorded, and the saved wave file was associated with different primitives. Generation of the feedback signals was based on a response time of 100 ms for haptic feedback and 28 ms for auditory feedback. Based on the maximum speed capability of the robot platform, the response time associated with generation of the feedback signal provided sufficient just-in-time information for user determination of robot movement (as discussed in the pilot

Fig. 4. Haptic interaction device used by individual student programmer.

study section). A description of these feedback primitives are given below:

1. Travel distance feedback: Feedback is triggered every few centimeters while the robot is moving in a forward direction. For our experiments, it is set to 10 cm, which is compatible with the size of the robot.
2. Turning left/right feedback: Different but symmetric signals are designed to provide feedback on robot turn status in fixed degrees. In this experiment, feedback is triggered every 45 degrees based on average robot speed and accuracy of odometry calculation.
3. Object distance feedback: An ultrasonic sensor, attached to the front of the NXT robot, detects obstacles between approximately 5 and 50 cm. Feedback is generated in fixed distance increments.
4. Bump feedback: When the robot collides with an obstacle, the mechanical system of the NXT robot triggers an exception. Feedback is associated with this condition in real time to provide the user immediate information about collisions.
5. Goal feedback: When the robot has reached a goal position, the goal event (triggered by the light sensor) is activated. This feedback information is also provided in real time to inform the user that the robot has successfully reached its final destination.

To correlate these feedback primitives to haptic and auditory sensory modes, we utilize two general-purpose devices. For providing haptic feedback, we utilize a Wii remote controller (Wiimote) (Fig. 4). The Wiimote is an interactive game controller that has several buttons for input and a motor for creating a vibration. This is a highly portable device and, with Bluetooth connectivity, allows serial communication with any paired computing platform. Haptic feedback is provided by coding functions that modulate the strength and duration of the Wiimote vibrations associated with the type of robot feedback needed. In order to create multilevel sensitivity for the haptic feedback, we control the force of vibration of the Wiimote by changing the on/off duty cycle of the vibrating motor using a pulsewidth modulation scheme [29]. Since the motor will continue to vibrate (with decaying order) even after the motor is cycled off, we used skewed Gaussian graphs to estimate the force envelope required for a single shot motor on/off. Using this approach, we can produce the desired sequence of vibrations and thus change the feedback force on the Wiimote. For controlling the pulse wave of the motor *f(t)* with a duty cycle D over duration T, we can estimate the output haptic force y as follows:

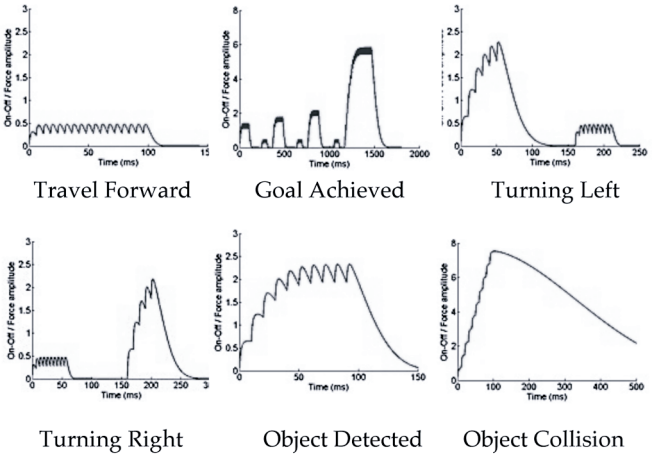

Fig. 5. Haptic vibration forces associated with robot actions.

$$f(t) = \begin{cases} y_{\max} = 1(on), & 0 < t < D \cdot T \\ y_{\min} = 0(off), & D \cdot T < t < T \end{cases} \quad (0 < D < 1), \quad (4)$$

$$\begin{aligned} \overline{y} &= \frac{1}{T}\int_0^T f(t)dt = \frac{1}{T}\left(\int_0^{D\cdot T} y_{\max}dt + \int_{D\cdot T}^{T} y_{\min}dt\right) \\ &= D \cdot y_{\max} + (1-D)\cdot y_{\min} = D, \end{aligned} \quad (5)$$

where y_{max} and y_{min} are output values of the system corresponding to the on/off states, which in this case are 1 and 0. Equation (5) implies that by controlling the duty cycle of a simple on/off system the system can, on average, produce a varying output using a pulsewidth modulation with duty-cycle D. Thus, by controlling the duty on/off cycle, we can change the strength of the feedback force on the Wiimote. This allows us to generate a series of vibrations associated with real-time sensory data, thus parlaying information about the robot actions and environmental characteristics (Fig. 5).

Auditory feedback is also associated with robot actions and is generated on the computer's speaker system. Various sounds associated with different piano notes were recorded, and the saved wave file was associated with action type (Fig. 6). Auditory feedback is provided by outputting the signals associated with the data correlated to the environment.

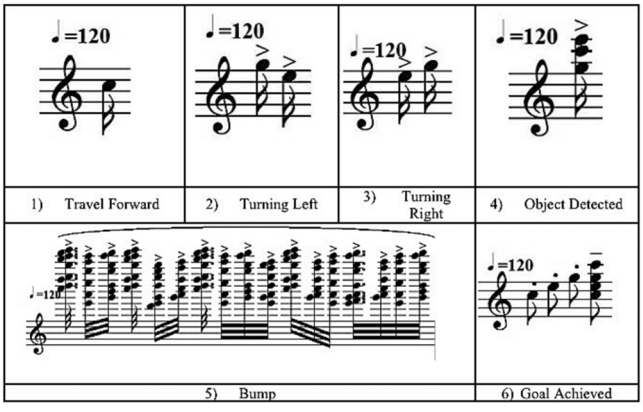

Fig. 6. Auditory feedback associated with robot actions.

CurTime	Sonar	Light	Touch	Touch	MotorA	MotorC	State	Deg(Reference)
1688	255	0	0	0	0	0	4	0
2100	255	0	0	0	0	0	4	0
2499	255	0	0	1	0	1	4	0
2905	255	0	0	0	-52	64	2	-26
3312	255	0	0	0	-104	101	2	-45
4550	255	0	0	0	120	122	0	0
5330	255	0	0	0	254	252	0	0
5741	255	0	0	0	249	249	0	0
6160	255	0	0	0	302	312	0	0
6928	255	0	0	0	455	461	0	0

CurTime	Sonar	Light	Touch	Touch	MotorA	MotorC	State	Deg(Estimation)
8383	76	0	0	0	0	0	4	0
8810	76	0	0	0	0	0	4	0
9259	76	0	0	0	23	-33	1	13
9655	25	0	0	0	68	-109	1	41
10948	9	0	0	0	44	54	0	0
11386	7	0	1	0	85	102	0	0
12004	6	0	1	0	90	110	0	0
12619	6	0	1	0	90	110	0	0
13236	6	0	1	0	90	110	0	0

Fig. 7. Robot log files: Top—scenario of successful task; Bottom—scenario of incorrect task solution.

As shown, all feedback primitives (travel forward, goal achieved, turning left, turning right, obstacle detected, obstacle collision) are designed for both haptic and auditory feedback, and are transferred to the user in real time while the robot is moving. Of particular interest, time durations for the haptic and auditory feedback signals were varied in duration, but minimized to the order of 100 ms to prevent interference between consecutive feedback signals. This is due to the fact that haptic feedback commonly requires longer time duration since haptic signals are perceived in low frequency and humans require more cognitive processing time for decoding of multiple sensory signals.

Using this form of feedback during program execution allows the students' to understand the robot movement during program execution, which provides them a means to associate their programming input to robot output.

3.4 Interaction/Feedback after Program Execution

Given the utilization of the haptic and auditory feedback devices, we also want to design a means to provide summary feedback information to the student once the robot had accomplished its programmed task. The purpose of this was to provide an automated way to evaluate the student's fully executed code. This would enable creation of a learning situation that does not completely rely on one-on-one teacher evaluation, but would enable students with a visual impairment to evaluate their own progress atomically.

The summary feedback process was "embodied" in an intelligent agent called Robbie which evaluated what the student's robot was "told" (i.e., programmed) to do (by the student) and provided audio feedback to the student after their program was executed. The voice of Robbie was provided by the text to speech functionality of the operating system.

Robbie was provided with "insight" by accessing the log file which contained the state estimation and sensing values created as the robot moved through the environment (and earlier discussed in Section 3.2). Robbie was written in C++ and used system calls to use the OS's speech functionality. The feedback provided is again dependent on the physical design of the robot, but is now also dependent on the log file that is generated while the robot operates.

In Fig. 7, we depict a snapshot of two example log files from a scenario where the task was successfully accomplished, and when an incorrect task solution was coded.

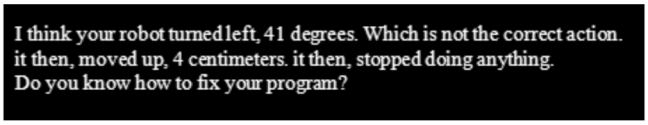

Fig. 8. Corresponding automated verbal response from log file.

Robbie's translation program (which automatically parses and translates information stored in the log file) associates the set of robot movement primitives (e.g., turn left, turn right, stop) with string descriptors (e.g., "turned left," "turned right," "stopped"}. Values from the motor encoders (MotorA and MotorC) are used to calculate values for the movement primitives, which are logged while the robot is executing the student's programmed set of actions. For each of the programming tasks, a reference movement primitive vector is also stored, which is compared to the student's robot output in real time. The verbal response from Robbie first relays information about the robot action; then compares it with the reference action. This is done in a repetitive manner until the log file associated with the robot's executed program has been fully parsed. Fig. 8 shows the corresponding voice response derived from parsing through the log file associated with the incorrect task solution.

4 Pilot Study and Evaluation

The accessible interfaces for programming were deployed and evaluated during a two-week summer camp held at the Center for the Visually Impaired. The programming sessions were attended by nine middle school students with various computer experience levels (Table 2). Mostly everyone had prior experience with a computer, but only a few had experience with programming. The goal of the sessions was to evaluate the effectiveness of the specialized modules to enable middle school students with visual impairments to participate in robotics-based computing activities and increase their interest in computing. To evaluate the interaction tools, we collected both objective and subjective measures. The number of trials to finish every task is used as the objective measure. In this case, a trial is defined as the full programming plus robot execution sequence needed to program the robot for the given task. For subjective measures, a survey was provided at the conclusion of the camp (Table 3).

Questions 1-2 and 4-5 were used to evaluate hypothesis 2—whether correlating haptic/audio feedback with

TABLE 2
Statistical Measures on Experience (Options Ranged from A Lot = 4; Some = 3; A Little = 2; None = 1)

	Average
Experience with a Computer	3.8
Experience using a gaming controller such as the Wiimote	3
Experience with programming	2.5
Consideration of working with computers or robotics when older	2.4

TABLE 3
Survey Questions from Pilot Study

1. How helpful was the Wiimote vibration for understanding the robot's movement?
2. How helpful was the sound feedback for understanding the robot's movement?
3. How helpful was it in determining what to change in your program based on what Robbie said?
4. Which do you think was most helpful for understanding what the robot is doing? (Wiimote, Robbie, both, none)
5. Which do you think was most helpful for knowing how to change your program? (Wiimote, Robbie, both, none)

real-time program execution was sufficient to enable the students to understand their programmed robot sequences. Questions 3-5 were used to evaluate hypothesis 3—whether verbal feedback to summarize program output (via Robbie) was sufficient for enabling the students to understand what changes had to be made to debug their program for correct output. Hypothesis 1—whether existing computer accessibility technology can be modified to provide sufficient feedback for students with visual impairments to enable the programming process—was evaluated based on the objective measure of number of task trials. Options for questions 1-3 represent a continuous scale with respect to the overall experience and ranged from very helpful = 4, helpful = 3, a little helpful = 2, and not helpful at all = 1. Questions 4 and 5 were used to provide a comparison metric between technologies and were represented by the options Wiimote = 4, Robbie = 3, both = 2, none = 1. Table 4 provides the associated mean response and standard deviation values for each question.

Based on the response from questions 1 through 3, the students agree that the haptic device was helpful in feeling the movement of the robot. But only slightly agreed that Robbie was helpful for determining how to change their program. With regards to questions 4 and 5, the majority (57 percent) felt that both Wiimote and Robbie helped in understanding what the robot was doing, with the majority

TABLE 4
Statistical Results from Pilot Evaluation

Options: very helpful = 4, helpful = 3, a little helpful = 2, and not helpful at all = 1		
Question*	Mean Response	Stdv
1	3.3	1.4
2	3.1	1.2
3	2.7	1.1
*options ranged from very helpful = 4, helpful = 3, a little helpful = 2, and not helpful at all = 1		
Question**	Mean Response	Stdv
4	2.9	1.9
5	2.7	0.7
** options were Wiimote = 4, Robbie = 3, both = 2, none = 1		

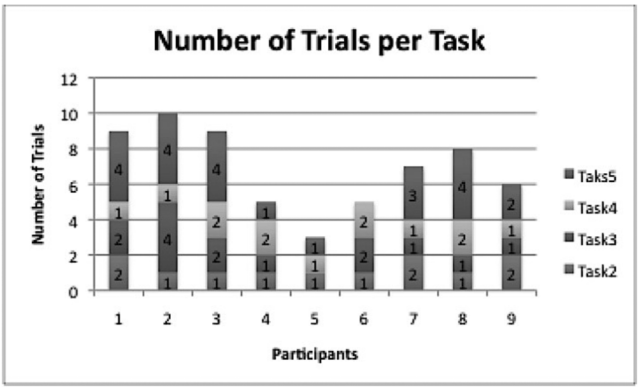

Fig. 9. Chart shows number of task trials.

(67 percent) stating that Robbie was more helpful than the Wiimote in determining how to change their program.

Fig. 9 depicts the number of task trials for each participant. These tasks are those described in Table 1. This data show that the modified utilization of computer accessibility technology (Hypothesis 1) to provide feedback/interaction via the integration of the screen reader, programming template, and standard BricxCC environment was sufficient to enable the students to program their robot. In most cases, the number of trials for successfully designing a correct solution increased based on the skill level required for completion of the robot task. Of particular interest was to evaluate the change in perspective of the students based on their experience. Table 5 highlights the change in perception (where options ranged from A lot = 4; Some = 3; A little = 2; None = 1). Based on pre-camp perception, there was slight agreement in the desire to work with computers/robotics, but with large deviation in the response. After participating in the robot programming sessions, it can be noted there was stronger agreement, with less deviation, among the students for them to consider working with computers/robotics as a career possibility.

5 Conclusion

In this paper, we discussed the use of alternative interface modalities to engage students with visual impairments in robotics-based programming activities. We provided an

TABLE 5
Measures on Perception

	Mean	Stdv
Pre-Session: How much have you considered working with computers or robotics when you grow up?	2.4	2.2
Post-Session: How much do you think this workshop helped show you that you are capable of working with computers or robotics?	3.4	0.8
Post-Session: How much has this workshop encouraged you to consider working with computers or robotics when you grow up?	3.4	1.2

overview of the interaction system and results on a pilot study that engaged nine middle school students with visual impairments during a two-week summer camp. The pilot study shows the feasibility of utilizing multimodes of interaction to enable students with visual impairments to participate in traditional robot programming processes. One key observation found is that the largest contributing factor to misunderstanding of robot movement was in the feedback signals that distinguish between left and right turns. We believe that this was caused by the symmetric nature of these signals and will be investigated in our future efforts. To further evaluate the impact of our methodology, additional sessions for students with visual impairments are planned for a longer period of time, such that we can measure their performance and learning curves on extended program tutorial tasks. Future work will also involve expanding the set of programming tasks in which the students are trained on, in improving the Robbie interface based on current feedback from the students, and in transitioning the interaction tools for use with open-source software. Ultimately, our goal is to provide a set of learning technologies that can be utilized by others to connect precollege level students with visual impairments to the computing world.

ACKNOWLEDGMENTS

This material is based upon work supported by the US National Science Foundation under Grant No. 0940146. The authors would like to express their gratitude to the Center for the Visually Impaired for advising on their robotic programming sessions, and to Rayshun Dorsey of WizKids Science and Technology Center for managing the camp logistics.

REFERENCES

[1] Nat'l Science Foundation, Division of Science Resources Statistics, *Women, Minorities, and Persons with Disabilities in Science and Engineering: 2002,* Sept. 2003.
[2] C. Mull, P. Sitlington, and S. Alper, "Postsecondary Education for Students with Learning Disabilities: A Synthesis of the Literature," *Exceptional Children,* vol. 68, no. 1, pp. 97-118, 2001.
[3] B. Bech-Winchatz and M. Riccobono, "Advancing Participation of Blind Students in Science, Technology, Engineering, and Math," *Advances in Space Research,* vol. 42, no. 11, pp. 1855-1858, 2008.
[4] US Census Bureau, Am. with Disabilities Act, July 2006.
[5] Nat'l Federation of the Blind Youth Slam, http://nfbyouthslam.org, Feb. 2011.
[6] The Alliance for Access to Computing Careers, http://www.washington.edu/accesscomputing, May 2010.
[7] Project ACE: Accessible Computing Education for Visually Impaired Students, http://www.se.rit.edu/~imagine-it/aboutus.html, May 2010.
[8] F. Michaud et al., "Assistive Technologies and Child-Robot Interaction," *Proc. AAAI Spring Symp. Multidisciplinary Collaboration for Socially Assistive Robotics,* 2007.
[9] Am. Printing House for the Blind, Inc., Annual Report, http://www.aph.org/about/ar2008.html, 2008.
[10] J.M. Francioni and A.C. Smith, "Computer Science Accessibility for Students with Visual Disabilities," *Proc. 33rd SIGCSE Technical Symp. Computer Science Education,* pp. 91-95, Feb. 2002.
[11] D. Tran, P. Haines, W. Ma, and D. Sharma, "Text-to-Speech Technology-Based Programming Tool," *Proc. Seventh WSEAS Int'l Conf. Signal, Speech and Image Processing,* pp. 173-176, Sept. 2007.
[12] K.G. Franqueiro and R.M. Siegfried, "Designing a Scripting Language to Help the Blind Program Visually," *ACM SIGACCESS Conf. Computers and Accessibility,* pp. 241-242, Oct. 2006.
[13] M. Calder, R.F. Cohen, J. Lanzoni, N. Landry, and J. Skaff, "Teaching Data Structures to Students Who Are Blind," *Proc. SIGCSE Conf. Innovation and Technology in Computer Science Education,* pp. 87-90, June 2007.
[14] R.F. Cohen, A.V. Fairley, D. Gerry, and G.R. Lima, "Accessibility in Introductory Computer Science," *Proc. 36th SIGCSE Technical Symp. Computer Science Education,* pp. 17-21, Feb. 2005.
[15] S.A. Ludi and T. Reichlmayr, "Developing Inclusive Outreach Activities for Students with Visual Impairments," *SIGCSE Bull.,* vol. 40, no. 1, pp. 439-443, 2008.
[16] J.P. Bigham et al., "Inspiring Blind High School Students to Pursue Computer Science with Instant Messaging Chatbots," *Proc. 39th SIGCSE Symp. Computer Science Education,* pp. 449-453, 2008.
[17] V. Kulyukin et al., "A Robotic Guide for the Visually Impaired in Indoor Environments," *Proc. Ann. Conf. Rehabilitation Eng. and Assistive Technology Soc. of North Am. (RESNA),* June 2004.
[18] I. Ulrich and J. Borenstein, "The GuideCane-Applying Mobile Robot Technologies to Assist the Visually Impaired," *IEEE Trans. Systems, Man and Cybernetics-A,* vol. 31, no. 2, pp. 131-136, Mar. 2001.
[19] C.H. Park and A.M. Howard, "Towards Real-Time Haptic Exploration Using a Mobile Robot as Mediator," *Proc. IEEE Haptics Symp.,* pp. 289-292, 2001.
[20] FIRST Lego League, http://www.firstlegoleague.org, 2010.
[21] D. Benedettelli, "Programming LEGO NXT Robots Using NXC," http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_tutorial.pdf, June 2007.
[22] Freedom Scientific, http://www.freedomscientific.com/products/fs/jaws-product-page.asp, 2010.
[23] A. Howard and G. Cruz, "Adapting Human Leadership Approaches for Role Allocation in Human-Robot Navigation Scenarios," *Proc. 11th Int'l Symp. Robotics and Applications,* pp. 1-8, July 2006.
[24] H.C. Waxman, M.-F. Lin, and G.M. Michko, "A Meta-Analysis of the Effectiveness of Teaching and Learning with Technology on Student Outcomes," Learning Point Associates, 2003.
[25] W. Yu, R. Ramloll, and S. Brewster, "Haptic Graphs for Blind Computer Users," *Proc. First Int'l Workshop Haptic Human-Computer Interaction,* pp. 41-51, 2000.
[26] C. Colwell, H. Petrie, D. Kornbrot, A. Hardwick, and S. Furner, "Haptic Virtual Reality for Blind Computer Users," *Proc. Third Int'l ACM Conf. Assistive Technologies (Assets '98),* pp. 92-99, 1998.
[27] S. Lee, G. Sukhatme, G. Kim, and C. Park, "Haptic Teleoperation of a Mobile Robot: A User Study," *Presence: Teleoperators and Virtual Environments,* vol. 14, no. 3, pp. 345-365, 2005.
[28] S.D. Laycock and A.M. Day, "Recent Developments and Applications of Haptic Devices," *Computer Graphics Forum,* vol. 22, no. 2, pp. 117-132, June 2003.
[29] G. Buja and G. Indri, "Improvement of Pulse Width Modulation Techniques," *Electrical Eng. (Archiv fur Elektrotechnik),* vol. 57, pp. 281-289, 1975.

Ayanna M. Howard is an associate professor at the Georgia Institute of Technology. Her area of research is centered around the concept of humanized intelligence, the process of embedding human cognitive capability into the control path of autonomous systems. This work, which addresses issues of autonomous control as well as aspects of interaction with humans and the surrounding environment, has resulted in more than 100 peer-reviewed publications in a number of projects—from scientific rover navigation in glacier environments to assistive robots for the home. To date, her unique accomplishments have been documented in more than 12 featured articles—including being named as one of the world's top young innovators of 2003 by the prestigious *MIT Technology Review* journal and in *TIME Magazine*'s "Rise of the Machines" article in 2004. She received the IEEE Early Career Award in Robotics and Automation in 2005 and is a senior member of the IEEE.

Chung Hyuk Park received the BS degree in electrical and computer engineering and the MS degree in electrical engineering and computer science from Seoul National University, Korea, in 2000 and 2002, respectively. He is currently a graduate research assistant and working toward the PhD degree at the Human Automation Systems Laboratory (HumAnS Lab.) in the Department of Electrical and Computer Engineering at the Georgia Institute of Technology. His research interests include robotics and intelligent systems, computer vision, human-robot interaction, haptic/multimodal systems, and assistive robotics. He is a member of the IEEE.

Sekou Remy received the PhD degree at the Georgia Institute of Technology's School of Electrical and Computer Engineering, where he was a member of the Human Automation Systems Laboratory. He is currently a Moreau postdoctoral fellow at the University of Notre Dame. Prior to Georgia Tech, he attended Morehouse College where he studied computer science and electrical engineering as a participant in the AUC Dual Degree Engineering Program. He is a member of the IEEE.