Project Title
Seeing Eye to AIBO:
Improving Human-Robot Interaction Using Virtual Reality and Voice Commands

Team Members
Ariel Rotter-Aboyoun, Nishanth J. Kumar

Abstract

A recent study conducted among AIBO users revealed the most-desired improvements were in the conversational and interaction skills of the robot, such as voice control. (Weiss et al., 2009; Kertész and Turunen, 2017) Little work has been done to extend the capabilities or usability of the AIBO, despite the user desire. To improve these features, we utilized new technology such as Ein, the ROS Reality Bridge, ROS Sharp and Google Voice AIY API. ROS Reality Bridge enabled us to link the AIBO with Unity and thereby create a Virtual Reality rendering of the robot that updates based on its actual position. Furthermore, we were able to use this to The Google Voice AIY API enabled us to process natural language, which is then mapped to commands and sent to the AIBO via ROS topics. Our work has enabled any number of AIBO's to perform basic actions via voice and motion commands (with the HTC Vive controller).

Additionally, we have created a Virtual Reality simulation of an AIBO that updates to the pose of a physical AIBO, which could enable future work such as interacting with AIBO's entirely in simulation or augmenting the AIBO to be able to fetch objects.

Introduction

The AIBO ERS-7 robot is intended to interact with humans (primarily children) to be a fun companion and instill an appreciation for robotics. However, current methods of interaction were limited to a few pre-programmed actions, such as response to touch, and text-based commands. The existing voice control on the AIBO is difficult to work with because it is written in the deprecated language URBI. Additionally, AIBO users' most requested improvement of the built-in functionalities was its conversational and interaction skills, including the voice control. (Weiss et al., 2009; Kertész and Turunen, 2017)

While there has been much work in the fields of human robot interaction through VR and speech, there has been very limited work to do so with dog-robots, specifically the AIBOs. The little work that has been done has been study or setting specific, for example, proving that a specific gesture-recognizing algorithm would work with the AIBO's camera. There has been no work that only aims to enhance user interaction with the AIBO.

As one step for improved interaction, voice control was implemented using Google's Cloud Speech API, interfaced with through the Google Voice AIY box. The output of the speech-to-text is processed by a client program on the Raspberry Pi, which sends corresponding Back commands to a server program on the laptop running Ein. The laptop publishes these commands and they're executed on the AIBOs.

Enabling interaction with the AIBO through motion input and VR required several steps. First, the AIBO's joint position and orientation needed to be streamed over a ROS Topic (/tf). Next, a program needed to parse this information and simulate a model of the AIBO using its URDF and associated 3D simulation files. Next, commands from motion controllers needed to

be recognized and mapped to specific commands to the AIBOs. These commands then needed to be sent to a specific ROS topic to actually command the AIBOs.

We evaluated our systems by attempting to command AIBO's through both interfaces and checking if they behaved as expected. Testing with common use cases, such as commands for sitting, standing, lying down, moving and turning were all successful, although we acknowledge that the mechanisms behind our control system can produce unexpected behavior outside of these use cases.

Related Work

Previous work to augment the AIBO's human interaction is limited, and has largely revolved around adding a few setting-specific features to the robot. For example, one group, working to make the AIBO into a good reading companion, made the AIBO turn towards sounds, read pre-recorded stories, or say pre-recorded encouraging messages. (Decuir et al., 2004) Another researcher added gesture recognition, but simply to prove that he could given the AIBO's camera. (Hasanuzzaman, M., et al., 2004) Published augmentations to the AIBO are either entirely experimental or not easily extended to functionality beyond the research's specific use for it.

Our goal to make the AIBO voice-controlled and viewable and interactable via VR would greatly increase the possible uses and extensions for the AIBO, as well as make existing interactions more user-friendly.

Technical Approach

To control the AIBO via the Google Voice AIY box, a translation must be done from English to URBI, the language that the AIBO is written in. As the Ein is easier to work with than URBI, we chose to have the voice box communicate with a computer running Ein and have the computer publish the commands it receives from the Raspberry Pi to the ROS topic that the AIBO is subscribing to. This communication is achieved through two Python programs, a client program running on the Pi, and a server program running on the computer. The sound received through the voice box's microphone is processed through the Google Cloud Speech API, which converts it to text. The Python code then searches for one of hotwords and, if it finds it, determines which of the preset Ein commands to send to the computer.

Controlling AIBOs in a dog pack, a group of multiple AIBOs can be done in one of two ways: iterating through the AIBOs with Back commands and sending back commands, or using a Back command to send the same URBI string to every AIBO in the pack at once. The voice control allows for either, but iterating through AIBOs is awkward as a verbal control system. The program also allows for summoning a single AIBO by name and having it perform a command, then resummoning the pack by keeping a queue of waiting commands in the client-side Python program.

The Virtual Reality aspect of the project was split into two distinct parts: enabling the AIBO's to be visualized in VR, and enabling them to be controlled with HTC Vive controllers. To accomplish the first part, the AIBO's joint state data was streamed onto the /tf topic using Ein to read the data directly from the AIBO. Scripts from the ROS# library were then used to connect to this topic via a subscriber, parse the data, and then update the 3D simulated model of the AIBO accordingly. However, the scripts did not use information from /tf about the orientation of robot in the world. Hence, the virtual model appeared rotated at odd angles.

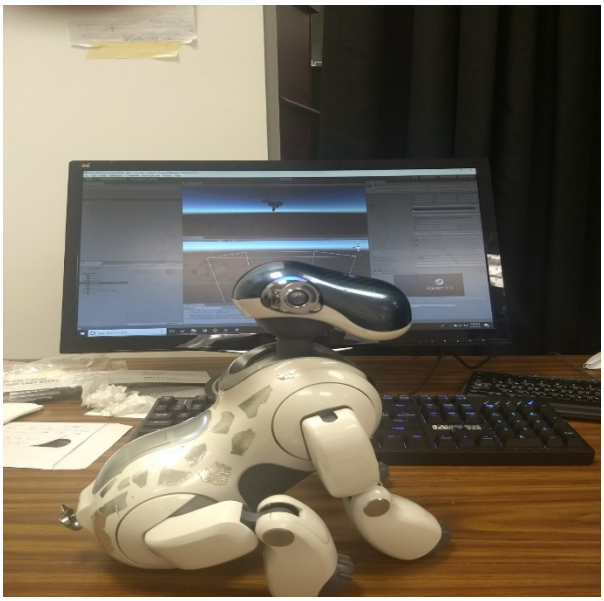

Fig 1: *Notice incorrect rotational orientation of virtual model on screen*

To resolve this issue, data about the robot's orientation (the position of the robot's base with respect to the world_frame) was streamed onto a separate topic called /ros_unity. This data was obtained from Ein, which itself used the AIBO's accelerometer and gyroscope to measure position and rotation. This data was read by another ROS# subscriber and the string was parsed to extract position and rotation data. The Unity simulation of the AIBO was then set to update accordingly.

Fig 2: *AIBO with corrected rotation*

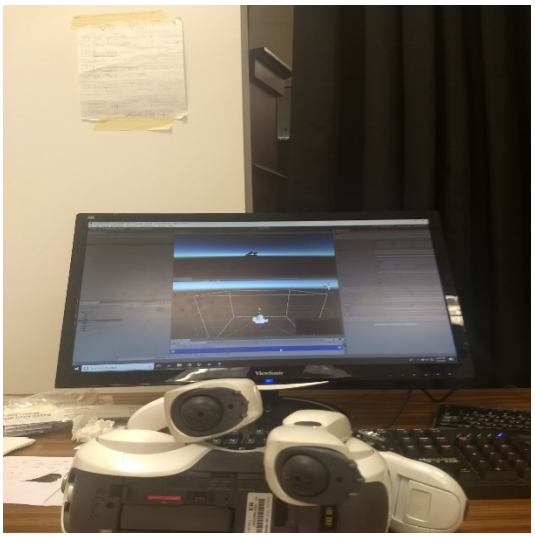

Fig 3: *No matter what angle the AIBO is rotated to, the virtual model is updated accordingly*

Data from the controllers was accessed via SteamVR (which is a plugin for Unity). A Unity script was written to map specific buttons and actions to commands for the AIBO (for example, holding the right-controller's trigger and moving the controller down was mapped to the AIBO being commanded to sit). These commands were then published to Ein via a ROS Topic (/ein/right/forth_commands). Once received, each command is simply executed in sequence by Ein.

Evaluation

For our final checkpoint, we defined the following goals:

- The AIBO model can be manipulated in VR to perform the basic commands (stand, sit, lie down, wag tail, lie down, turn right and left, and heel).
- Changes to the joint angles of the physical AIBO will update the joints of the virtual AIBO
- Several AIBOs can be voice controlled to perform the basic commands simultaneously or one referred to by name to perform an action.

All goals were partially met. The AIBO can be controlled via buttons on the VR controllers to perform most of the commands, but the "heel" command was never created for the AIBO. This is because 'heel' required data about the AIBO's absolute position in the room, which was too difficult to obtain without a camera specialized to the task (or some other dedicated localization hardware).

The AIBO in VR updates when the physical AIBO changes position/orientation, but the physical AIBO doesn't read the changes in its joint angles while changing position so the virtual AIBO will jump to the new position after the physical AIBO finishes moving.

Commanding multiple AIBOs is supported in the existing code and the voice control system uses these preexisting commands. However, the main way of doing this is by iterating through the AIBOs and having each perform a command individually. The only commands supported for simultaneous movement are stand, sit, and lie down. If a command besides these is

given in pack mode, only the single AIBO who is the current “focused member” will perform it. Commanding a single AIBO by name works and any commands can be used in this mode.

Conclusion

For our project, we decided to extend the ease of interaction of the AIBOs by adding new control options for it: voice control and control through VR. We’ve had partial success with in all of our end goals and realized many of the limitations of the AIBO system in the process, but we have implemented voice control for a one or more AIBOs and made a single AIBO viewable and controllable in VR.

The largest issue with adding more commands for multiple AIBOs is the intricate way in which the Ein, C++, and URBI interact with each other. If the AIBOs are controlled with Ein’s back commands, then steps in the command, such as reading joint angles, will block other commands from being executed. This means the AIBOs will perform commands one after another. To make them perform commands in unison, URBI strings must be sent directly. This allows the AIBOs to stand, sit, and lie down in unison. However, many of the movements and all of the intermediary steps, like getting joint angles, are written in C++. For example, while the dog pack can all play the barking noise in unison, they cannot open and close their mouth at once. Making the AIBO pack capable of any movement a single AIBO can perform would require large refactoring of the C++ code, which was out of the scope of this project but a possibility for future work.

There are several ways the VR interaction system could be extended in the future. As mentioned above, one could incorporate some new localization hardware to track the AIBO’s absolute position in a space and allow it to perform tasks like fetching and heeling. Furthermore, our system only supports one AIBO model in VR at a time because the /tf topic only transmits the joint states of one AIBO at a time. In the future, a workaround could be made to allow multiple AIBO’s to exist in the same scene and be updated individually. Another extension would be to simulate environments and tasks for the virtual AIBO and have the real AIBO mirror those actions, so one could perhaps play games with the virtual and real AIBOs.