Voice Interfaces for Human-Robot Interaction

1st Rebora, Davide
EMARO+
Genoa, Italy
davide.rebora@gmail.com

2nd Calvo, Natalia
Robotics Engineering
Genoa, Italy
nata.calvob@hotmail.com

***Abstract*—Voice user interface has became one of the most natural and intuitive methods of human-machine interaction. This document presents a model for speech recognition implemented for the Baxter Robot. Using cognitive interaction tasks provide by different companies as Microsoft, Google between others; it is possible to build an interface which allows to the user give voice commands to the robot in order to execute different actions. The speech recognizer implemented searches the first grammar to matching key words for the utterance using context specific grammars. Besides, the user has the option to build his own grammar storing the adaptive updates into the User Module, in order to enhance speech recognition processing efficiency. Due to this fact, users can easily access the adaptive words and modify them according to their requests, shaping the interface to meet the demand of every specific user.**

***Index Terms*—Human robot interaction, speech recognition, Voice Robot Interface.**

I. Introduction

Human computer interaction opens the door to create systems that may assist humans in a large variety of fields. Using natural language interfaces it is possible to create modules which allow humans to give voice commands to the robot, and then to the robots to pro actively act to the human requests and actions. Indeed one of the goals of speech recognition is to allow natural communication between humans and computers via speech, where natural implies similarity to the ways humans interact with each other. This document pretends to explain the interface build for the Baxter Robot, using the speech recognition interface provided by Google. The aim of this work is to investigate the potential application of a speech recognizer and design a module where the user can give a large number of instructions to the Baxter using voice commands to control it. The robot's decision making is based on interpreting voice string and processing it among a series of conditional cycles in order to provide to the user the required task executed by the robot. Moreover the interface is built by following the main criterion of human computer interaction as learnability and usability

In the following chapters, there will going to discuss more about the system, introducing a Voice Robot Interface between the user and the robot for interaction purpose. This document starts with the explanation of the literature review about the speech recognizers tested. Then next section is related to the design of the voice user interface and the implementation of it to control the robot's actions. Finally, there are the conclusions and the limitations as well as future work.

It is important to remark that the aim of this work is to build a voice interface and not a natural language processor.

Group Project. EMARO+ Università degli Studi di Genova

II. Speech Recognizers

Speech Recognition technology promises to change the way humans interact with machines in the future. This technology is growing day by day and roboticists are working harder to overcome the remaining limitations. Speech recognition is the process by which a computer identifies spoken words. Basically, it means that an user can talk to the computer, and having it correctly recognize what he is saying. There exist two main tasks in Speech Recognition: recognize the series of sound and identify the word from the sound. This recognition technique depends also on many parameters - Speaking Mode, Speaking Style, Speaker Enrollment, Size of the Vocabulary, Language Model, Perplexity,Transducer [1] etc. There exists many speech recognizer API's on the market that can be incorporated on different applications. Due to the increasing number of these systems it is not easy to identify which application has a better performance, however this section is dedicated to analyze the advantages and limitations among some of them. For this report, there was selected three commercial open-source systems: Google Speech API, Microsoft Bing Voice Recognition and CMU Sphinx.

A. Google Speech API

Google has developed the Speech Recognition API which enables easy integration of Google speech recognition technologies into developer applications. Using this API, developers can create products and services transcribing the text of users who speak into a microphone. Google can recognize over 80 languages and variants. Recently, Google has implemented machine learning mostly deep learning neural networks in order to improve the accuracy of its technology reducing the error rate.

There exist two API's which can be implemented to work on-line, the Google Speech Recognition and Google Cloud Speech. The last one is a paid version, its performance is better and also provides support through a set of cloud-based interfaces. Indeed the implementation of neural networks allows to this application improves the speech recognition in real time.

On the other hand, the free version of Google also has a good

performance and it is easy to implement, however the accuracy and robustness is reduced with respect to the paid version.

B. Microsoft Bing Voice Recognition

Microsoft Bing Voice Recognition is a speech recognition technology that transcribes audio streams into text. It is also available as an API that Microsoft makes available to software developers. However, it generates a limitation, because it is needed an Microsoft account in order to implement this interface. This support only is offered for 15 languages, but it can work in different scenarios, interactive, dictation and conversation. Microsoft has a free trial that has a limitation of requests per month.
According with many researches, the advantage of implementing Microsoft Bing Voice Recognition is that it is possible to implement a large-vocabulary in order to reach human parity with conversational speech recognition [3].

C. CMU Sphinx

The Sphinx system has been developed by researches of Carnegie Mellon University, and it is writing in the Java programming language, which has inherent advantages from the stand-point of code maintenance. Sphinx has a large vocabulary and a speaker independent speech recognition code-base. It is highly modular and flexible, supporting all types of HMM-based acoustic models, all standard types of language models, and multiple search strategies.The code can be downloaded by free and also it allows to work off-line. This system has tree different models, acoustic model, the phonetic dictionary and the language model, which are combined together in order to generate a speech recognizer.

III. Voice Human Robot Interface

User interface is an important component of any product handle by the human user. The concept of robotics is to make an autonomous machine, which can either replace human labor or work in cooperation with human. However, to control the robot or to provide guide line for work, human should communicate with the robot and this concept led the Roboticist to introduce User Interface to communicate with robot. Furthermore, for humans, the most intuitive way to interact is by speech. Under this concept, the Voice Human-Robot Interface -VHRI- was designed under the free trial provided by Google using simple grammar sentence instructions, like “Move”, “Go”, “Left”,"Close", and so on. After to make several tests with the systems described in the previous chapter, the Google Speech Recognition API is the one which gives better features according with the requirements of the VHRI with an acceptable accuracy of the utterances.
The aim of this interface is to reduce the gap between humans and the way to give instructions to the robot for a specific task or action, using voice commands to control the Baxter. The VHRI has two main sections as is despited in the block diagram of the figure 1. The following subsections explain in detail each of the blocks and their functions.

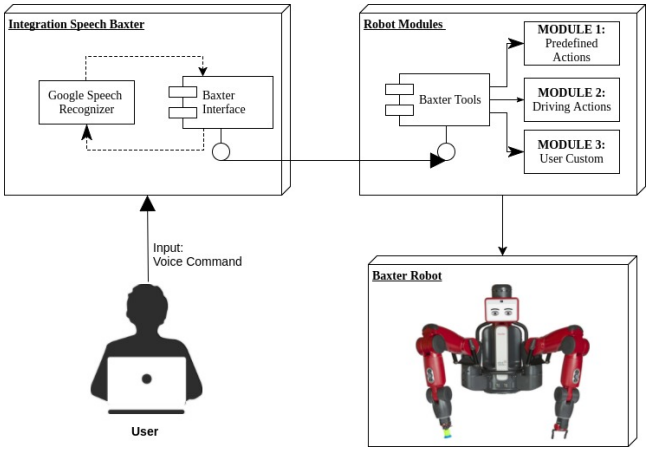

Figure 1. Voice Robot Interface Diagram

A. Integration Speech Recognizer Baxter

This part is related with the initial state of the VHRI, it means the capture of the voice commands from the user, the processing of the signal and the integration to the Baxter Interface.
The speech commands are taken to the computer by the microphone, this signal is processed by the Google Speech Recognition which converts the wave signal into text in order to understand which is the specific action that the user wants to perform. Once these data are obtained, the following step is to match the commands provided with the key words in order to enable the robot and arrange it to perform the required tasks. If the program is not able to understand the voice commands at the first time, it will ask the user to try again as many times it requires until it finds the words which match with one of the command.
If the interface can recognize the correct command by the speech recognition system, it converts the voice command into the enable action recognizable by the Baxter and then the robot would get the enable command and executes the action, this fundamental tool allows for enabling/disabling/resetting/stopping the robot.
Besides, the module has the capability to give a feedback to the user using also speech. In order to obtain this feature, it was implemented the Google Text to Speech library, this is important due to the fact that the user can receive the status of the task, if the action has success or if it necessary to give a new command, among others.

B. User Modules

Three modules have been designed in order to increase the usability and the learnability of the Interface, and also to interact and control the robot in different scenarios. The user can switch between these modules by using a voice command. At the beginning, in order to start the program, the voice command*Hello Tony* is required, which enables the motors of each joint of the robot. Once the Baxter is enabled,

the interface asks the user which module wants to use, that strictly depends on the specific work that the consumer needs.

1) MODULE 1: Predefined Actions: Baxter has a large number of actions that can be executed using ROS packages. These tools are provided in the baxter_tools package, and are largely convenient, common usage and executable tools created using the baxter_interface.
There exist many predefined actions which can be ordered by the user, inside the ROS environment are so-called services, their functionality is to build directly upon the ROS msg format to enable request/response communication between nodes. Keeping into account that the main interface of the Baxter RSDK is via ROS Topics and Services using core information needed to interface with Baxter, these services can be controlled by the user using the proper key word[1]. The aim of this module is to allow the user to perform several standard tasks already presented in the robot packages. Some of these standard actions are frequently used pointing out the usefulness of this module. Following is a brief explanation of how the user can interact with this module.
One of the most common action is the Tuck Arms Tool, it is a tool for tucking and untucking Baxter's arms. Useful when unpacking Baxter, moving locations, shipping, etc. This action can be executed giving the voice command *ready* or *work* to the VHRI. On the other hand the tuck movement can be executed by giving as input *goodbye*. Another common task that has been implemented into the interface is the opening and closing of the gripper, which can be done by specifying the hand for instance *open right hand*. It can be also used the word *gripper* in place of *hand*.

Furthermore, the VHRI can return to the user if the task had success or not during the execution, lets it know the real state of the task. In order to achieve this goal, the VRI has a function callback which has the purpose of store the message that the action generate during the execution of itself. When the order is given by the user, the callback return the current message with the information of a specific state of the robot. From the feedback point of view two listeners have been developed which subscribe to the topics *robot/state* and *robot/end_effector/right_gripper/state* by importing from *baxter_core_msgs.msg* respectively *AssemblyState* and *EndEffectorState*. The former one is used for checking if the robot is enabled correctly while the latter for being sure that the gripper is closed or open correctly.

On the other hand, there exist some actions that can be executed as a client action using a server action. Using the **actionlib** package which provides a standardized interface for interfacing with tasks. The user can implement it to send goals for the Baxter, for example, to pick one object, move the arms to the desired position, and so on.
Correlated with the developed of the module, it is considered necessary to explain some details. The actions and servers are subprocesses that must be executed parallel with respect to the main process of the VRI. In order to handle this mechanism it was used the subprocess module, which provides a robust interface to create and work with additional processes.

2) MODULE 2: Driving Actions: This module was designed to provide to the user the option to control the Baxter's movements directly. It means that the user can manage the basic joint position using simple voice commands. Baxter has 7 joints (DoF) in each of its two arms as it is depicted of figure 2. There are two joint for the shoulder, two for the elbow and three for the wrist (S=Shoulder, E=Elbow, W=Wrist). The user must to familiarize and identify each joint to be able to perform the proper movements. The Baxter's module implemented was the controlling of the Position Mode, the idea was to publish target joint angles for given joints and the internal controller drives the arm to the published angles, where the units for the position of a joint are in (rad). However for this application the user will manage each command in degrees, resulting more intuitive for humans. First the user has to give the command *Driving* to enter to this module. Then he can choose between the right or left arm in order to give the instructions. Each joint has a unique key work to enable it[2]. The joint does it move changing the value of the angle, the step to increase the angle is $5degree$. Once the desired joint is selected, the trajectory is executing with the command *Go*, it means that the joint will reach the value set on the default step size value, if the user notices that the position reached is not the desirable, he could increase the angle of the joint with the command *Increase*, which simultaneously execute the action **joint_trajectory** to achieve the new position. Besides, it is possible to change the direction of the rotation using the command *Change*. Furthermore, the user can stop the driving action when it has achieved the goal of its task saying *Stop*, at the same time it can store the value of the angle final position of each joint with the command *Save*. There exist three slots in which the user can store the final position of the joints, also those can be chosen given the proper command.
Other useful feature of this module is the option to open or close the end effector of each arm using the voice command *Open* or *Close*. After the user has selected the right or left arm, using the command *Gripper* it can select the corresponding end effector. Finally the user can exit of each mode with *Exit*. This data stored could be useful to the user to generate its own actions according to its needs in the Module 3 so called **User Custom**.
As a remark, the user must to know which voice command corresponds to a certain joint, keeping into account that there are fourteen possibilities to control the robot. It has been provided a user commands guide in the appendix of this document.

[1]See Appendix.

[2]See Appendix

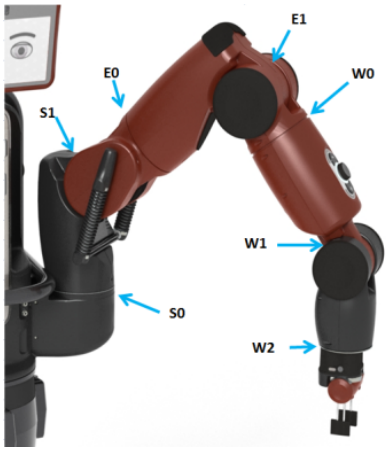

Figure 2. Named Joint Mapping - Baxter's Arms

3) MODULE 3: User Custom: Last but not least, it was designed a third module which allows to the user create and generate its own actions or tasks, and at the same time gives the possibility to add more key words in accordance with the tasks that he wants to perform. To access to User Custom module the command is *User*. Once the user has started with this module has the option to select two different modes of work, e.i. *Play* command enables the section in which the user can perform its own actions, parallel with the *Create* command the user can record the robot movements by moving it manually in order to generate a text file which stores the desired joint trajectory.

This module designed in a high level can be integrated in three different approaches. The first approach works under to the assumption that the user assigns a **goal position** to the Baxter, and gives the order to reach the goal. It was designed regarding the feature provided by the module 2, where the user can store the final position of a driving action, this data is taken at this stage as the goal position for tracking trajectory execution. The command to execute this section is *First*, then the user choose the slot in which the data has been store with the respective command, it sends the message to the robot to reproduce the action beginning to follow its trajectories in order to achieve the goal. As a remark, the initial posture of the robot is the untuck position.

The second approach is called with the voice command *Second*, the VHRI was developed in such way that the user can perform its own actions according with its preferences. Extending this concept, the user could be add the executables of the actions to a text file, following the next structure key_word_1 key_word_2 ; shell_command. for example tuck arms; rosrun baxter_tools tuck_arms.py -u. , remark that the **";"** and **"."** are compulsory in this structure. In this way each action is assigned to key words choosen by the user in order to add up such words to the grammar of the VHRI, hence the interface matches the key words to the user's actions and send the respectively command to the robot in order to execute the action desired. This feature points out that the interface has been built in order to be customisable to satisfy as much as possible the demand of any kind of user.

Finally, the third approach activated with the command *Create* was developed in the interest of implement tasks directed by the user using demonstration form. It means that the user can enable the robot joint trajectory interface to record the joint position with corresponding time stamps using the voice command *Start*, then Baxter starts to record the positions of the both arms and the gripper actions (open or close). Once, the user has finished moving the robot, he says the proper command to end the record

textitStop and generate a text file which stores all the positions achieved during the record. In order to playback the joint trajectory, it is necessary to run the corresponding controller, this is possible executing the joint trajectory action server provided by the baxter_interface implementation to support the action using the command *Third*. The user has to send this command in order to open a new shell and execute the subprocess. Then, the playback action parses the file created using the joint position recorder and sends the resulting joint trajectory to the action server in order to be reproduce the actions by the Baxter. There exists the possibility to reproduce more than one directed action, just selecting the adequate command.

IV. Conclusions

The Interface has a good performance, has the capability to interact with the user simulating a conversational environment, receiving the voice commands from the user, and giving the feedback by speech. The three modules were tested in order to evaluate the accuracy of the interface, the time of response and the error rate. Under these results have been concluded the following considerations.

The Voice Robot Interface has a dependent feature, it means that the user must train the commands before starting interact with the VHRI, those commands are largely explained in the Appendix. Besides, the speech recognition is sensitive to noise but also sensitive to voice tone change in order to recognize the sentences. During the experiments when some key words have similarities in pronunciation and also syntax, the speech recognizer can not identify the proper word. For this reason, the commands selected are simple and easy to identify. However in ideal conditions (not noise and neutral tone) the interface can recognize complex sentences producing a conversational environment with the user.

With respect to the time of execution, it is directly proportional with the complexity of the task that the user wants to perform, but also it depends of the complexity of the sentence given by the user, keeping into account that for this application was used a free trial, when the speech is large and complex the recognizer could tend to failure, and the time of response increases. Furthermore, the implementation of the library gTTS to reproduce speech from the interface produce delays in the response.

The Voice User Interface was designed in such way that it

could be adaptable to the requirements of a user using a looped structure. It is an important feature from the computational effort point of view, because the structure of the interface is based on cycles that enable or disable the actions in accordance with the command given, if the command does not match with the key words corresponding to a certain action, it disables that action and continues until finding the proper matching. Due to the fact of the large complexity of this model, this methodology provides a good performance and avoid the dumped of the system.

V. Further Work

- The using of Natural Language Processing can increase the accuracy and robustness of the interface, in order to avoid the rate error and improve the intent of the speech of the user. The speech recognition interface implemented during this work returns text results, each result with a certain level of confidence. NLP can takes this data and extracts a meaning representation through Natural Language Understanding that allows the creation of conversational voice interfaces for different applications in the robot.
- The Voice Robot Interface can be improved in so many ways in order to augment the capabilities to give commands and execute actions. For instance, it is possible to introduce more complex activities and sentence to the interface.
- Other interesting field will be to introduce gestures which are important part of the Natural Language. Due to the fact that humans generally use gestures to interact with another human under a normal conversation, either to pointing to an object or to give some indication with the spoken language. There are also researches going on to introduce speech recognition interface with gestures recognition, this interface called multi-modal communication interface.

References

[1] Survey of the state of the art in human language technology. Cambridge University Press ISBN 0-521-59277-1, 1996. Sponsored by the National Science Foundation and European Union, Additional support was provided by: Center for Spoken Language Understanding, Oregon Graduate Institute, USA and University of Pisa, Italy,

[2] P. Lange and D. Suendermann, Tuning Sphinx to Outperform Google's Speech Recognition API, The Baden-Wuerttemberg Ministry of Science and Arts as part of the research project. .

[3] Microsoft Corporation (2016) Exploring New Speech Recognition and Synthesis APIs In Windows Vista.

[4] A. Waibel and K. Lee. Readings in Speech Recognition, ISBN 0080515843, 9780080515847, Elsevier, 1990.

User Command	*Robot Action*
Integration Speech Baxter	
├─Hello Tony	Enable the Robot
│ └─Standard	Select Module 1
│ ├─Ready *or* Work	Untuck the arms
│ ├─Open	Open Arm or Gripper
│ ├─Hand *or* Gripper	Open the Gripper
│ ├─Right	Open the Right Gripper
│ ├─Left	Open the Left Gripper
│ ├─Arm	Open the Arm
│ ├─Right	Open the Right Arm
│ ├─Left	Open the Left Arm
│ ├─Goodbye	Tuck the Arms and disable the Baxter
│ └─Driving	Select Module 2
│ ├─Right	Select Right Arm
│ ├─One	Select joint right_w2
│ ├─Go	Rotate the joint right_w2
│ ├─Increase	Increase the angle of joint right_w2 in 0.1rad
│ ├─Change	Change the orientation of joint right_w2
│ ├─Stop	Stop the movement and exit
│ ├─Two	Select joint right_w1
│ ├─Three	Select joint right_w0
│ ├─Four	Select joint right_e1
│ ├─Five	Select joint right_e0
│ ├─Six	Select joint right_s1
│ ├─Seven	Select joint right_s0
│ ├─Gripper	Select joint right gripper
│ ├─Open	Open the right gripper
│ ├─Close	Close the right gripper
│ ├─Save	Store the Final position
│ ├─One	Store the final position into the Slot 1
│ ├─Two	Store the final position into the Slot 2
│ ├─Three	Store the final position into the Slot 3
│ ├─Exit	Exit of the Module 2
│ ├─Left	Select Left Arm
│ ├─One	Select joint left_w2
│ ├─Go	Rotate the joint left_w2
│ ├─Increase	Increase the angle of joint left_w2 in 0.1rad
│ ├─Change	Change the orientation of joint left_w2
│ ├─Stop	Stop the movement and exit
│ ├─Two	Select joint left_w1
│ ├─Three	Select joint left_w0
│ ├─Four	Select joint left_e1
│ ├─Five	Select joint left_e0
│ ├─Six	Select joint left_s1
│ ├─Seven	Select joint left_s0
│ ├─Gripper	Select joint left gripper
│ ├─Open	Open the left gripper
│ ├─Close	Close the left gripper
│ ├─Record	Store the Final position
│ ├─One	Store the final position into the Slot 1
│ ├─Two	Store the final position into the Slot 2
│ ├─Three	Store the final position into the Slot 3
│ ├─Exit	Exit of the Module 2

User Command	*Robot Action*
Integration Speech Baxter	
├─Hello Tony	Enable the Robot
│ └─User	Select Module 13
│ ├─Play	Play the user's actions
│ ├─First	Reproduce an action recorded in the Driving module
│ ├─One	Reproduce action stored in slot one
│ ├─Two	Reproduce action stored in slot two
│ ├─Three	Reproduce action stored in slot three
│ ├─Second	Reproduce a user's action inserted by Text File
│ ├─One	Reproduce the user's action one
│ ├─...	...
│ ├─Five	Reproduce the user's action five
│ ├─Third	Reproduce a directed movement by the user
│ ├─One	Reproduce the user's action one
│ ├─...	...
│ ├─Five	Reproduce the user's action five
│ ├─Create	Enable the robot to the record mode
│ ├─Start	Start to record the directed movements
│ ├─Yes	Save the directed movements in a text file
│ ├─One	Save the directed movements in the first text file
│ ├─...	...
│ ├─Five	Save the directed movements in the first text file
│ ├─No	Do not aave the directed movements and exit
│ ├─Stop	Stop to record the directed movements