 DOI: 10.3217/978-3-85125-663-5-16

Multilingual Speech Control for ROS-driven Robots

Dominik P. Hofer[1], Simon Brunauer[2] and Hannes Waclawek[3]

***Abstract* — To improve the collaboration of humans and robots, a multilingual speech (MLS) control was created, which allows to manage multiple ROS-based robots at any time.**

Keywords—Speech Control, Multilinguality, Speech to Text, Text to Speech, Intent and Variable Detection

I. Introduction

How can various types of Robots (e.g. robot arm, robot car, etc.) be controlled with the same speech control? In this paper, a universal speech control for various robots, which use ROS as a middleware, is presented. The speech control only uses open source tools and libraries which work offline. This allows the user to secure privacy and independence of companies. Furthermore, to allow people with different mother tongues to use the same speech control, multilingualism was implemented.

II. Method

The workflow for the whole audio dialog was split into five steps: First, language identification (LID), next speech-to-text transformation (STT), afterwards the intent and the variables of the command are detected (I&VD), which is followed by command publishing (CP) and finally the creation of the audio response by text-to-speech (TTS). Figure 1 is a graphical representation of the speech control. For the implementation of this workflow, a set of specialised open-source tools was used, which are described in the following subsections.

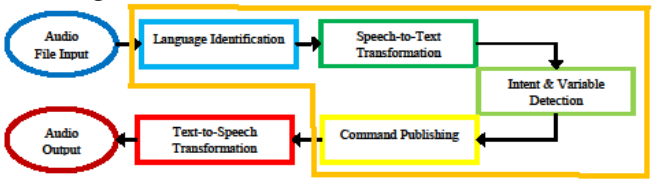

Figure 1: The graphical representation of the processes done by the speech control

A. Language Identification (LID)

For the speech control to transform the audio signal into the correct written language, the LID is necessary. In the case of this project the library iLID, created by Tom Herold and Thomas Werkmeister, is used [1]. The library allows to train a convolutional neural network (CNN) with Mel-Filters on various languages [1].

B. Speech to Text (STT)

The STT-Transformation is done by using the CMU Sphinx library [2]. This open-source offline tool allows to transform any language by letting users create their own dictionaries, language models and acoustic models.

C. Intent and Variable Detection (I&VD)

To let the system know, what the user wants a specific robot to do, the intent, as well as variables, need to be extracted out of the command. ID detects the intent the user has e.g. drive, bring, stop, etc. SF, on the other hand, is the process of finding variables in the command (e.g.: "*fahre 15 Meter*" – 15 meter is the variable). This is achieved by using the Snips NLU tool, which analyses the text via regular expressions or using a logistic regression in combination with Conditional Random Fields (CRFs) [3].

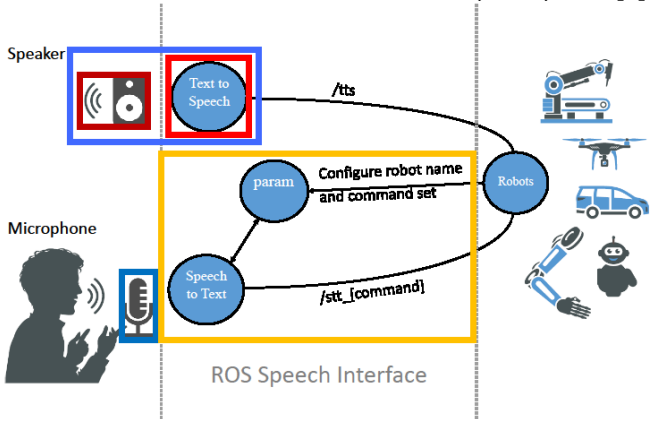

Figure 2: Architecture of the ROS Speech Interface

D. Publishing Command (CP)

For the robot(s), referenced in the audio file, the intent and the stated variables are published on a topic. Two examples of ROS topics can be seen in Table 1. Column one shows the name of the topic. The message format stays the same for all the topics and consists out of three strings. The first string representing the name of the robot. The second referencing the language which was identified by the ID. This was implemented to allow the Text to Speech-Transformation to

ROS Topic	ROS Msg format	Intended Function	Usage Example
/stt_save_pos	stt_std	Save current robot position under a position name	*"Sam, speichere diese Position als Startposition."*
/stt_goto	stt_std	Go to position previously saved via /stt_save_pos	*"Sam, fahre zu Startposition."*

Table 1: Some ROS Topics and the corresponding functionalities

[1] Dominik P. Hofer is with Salzburg Research Forschungs GmbH, 5020 Salzburg, dominik.hofer@salzburgresearch.at and Information Technology & System-Management, Fachhochschule Salzburg GmbH, 5412 Puch bei Hallein, AUSTRIA, dhofer.its-m2018@fh-salzburg.ac.at

[2] Simon Brunauer is with Information Technology & System-Management, Fachhochschule Salzburg GmbH, 5412 Puch bei Hallein, AUSTRIA sbrunauer.its-m2018@fh-salzburg.ac.at

[3] Hannes Waclawek is with Information Technology & System-Management, Fachhochschule Salzburg GmbH, 5412 Puch bei Hallein, AUSTRIA sbrunauer.its-m2018@fh-salzburg.ac.at

respond in the correct language. The final string consists out of the function the robot should execute. The third column of Table 1 shows what function the topic sets in motion. The final column represents sample sentences.

E. Text to Speech (TTS)

The final step, the TTS-Transformation uses the response (further explained in chapter III. Results) of the robot and transforms it into an audio file. This is done by the eSpeak library [4]. This library allows to transform text written in various languages.

III. Results

The architecture of the speech control as shown in Figure 2 consists out of four different ROS nodes: *stt*, *param*, *client* and *tts.*

The *stt* node focuses on the entire process of speech and language analysis, as well as checking if the intended command is supported by the demanded robot. The *param* node, strictly speaking is already integrated in the ROS core node, is a parameter server which holds a list of all the robots currently active in the ROS environment and their supported commands. The parameter server does this process automatically. The *client* node represents the various robots. The final node, *tts*, only focuses on audio output and listens on the tts-topic. The node saves any kind of message in a first in first out (FIFO) concept.

For the implementation of the speech control, a demo scenario was created. This scenario is represented as a dialog flow chart in Figure 3. It shows how a user asks the robot for a certain task. After that the speech control evaluates the command, whether the robot can even execute the task, or if everything is fine. The error analysis is done multiple times during the execution. If the user wants the execution to stop, the stop command can be executed at any given moment. This allows handling of dangerous situations, but it is still recommended to have further security tools at hand.

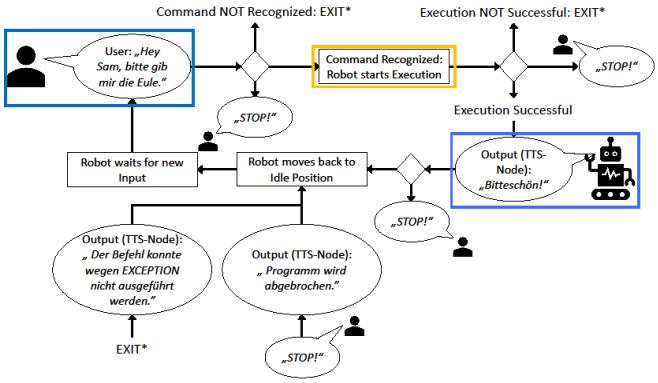

Figure 4: Dialog Flow representation of the Proof-of-Concept

If everything went according to program, the speech control outputs an audio message and the corresponding robot moves back into the idle position. Figure 4 shows an implemented proof-of-concept. Here, the Panda Robot by Franka Emika [5] transports a product (3D-printed owl) from the storage (not in the picture) to the product output facility (conveyor belt).

IV. Further Work

There are some possibilities of improving the speech control. The first and probably most important one would be to allow different accents and dialects. This would provide users an even better experience. This can be achieved by focusing on two points regarding the Speech to Text transformation. The first one would be to allow the use of certain English commands in other languages. This would diminish the need of creating an e.g. German counterpart for e.g. "gripper". The second improvement would be to advance the resources available for CMU Sphinx. This would allow to better use a bigger variety of languages. Another improvement would be understandability. This should mean that it can take several tries till the speech control got the correct command. This problem can also be solved by improving the stt transformation. The last improvement would be speed. The process of analysing the audio input until the robot finally moves currently can take up more than ten seconds. This should be reduced in further research.

V. Conclusio

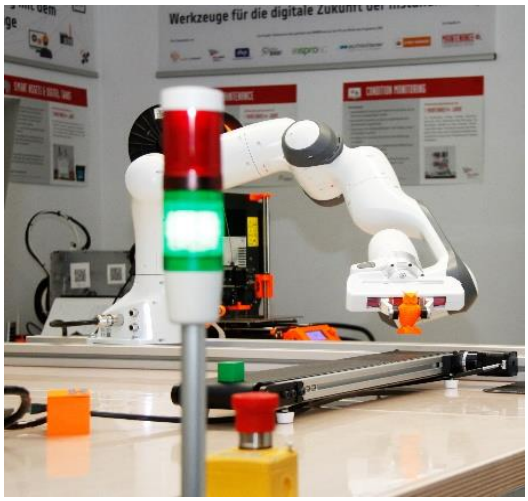

Figure 3: Implementation of the Proof-of-Concept

All in all this short paper presents a speech control that allows to command multiple robots, which have a ROS interface, in multiple languages. The libraries and tools used are on the one hand open-source, which makes the user company independent, and on the other hand do not need an internet connection. This aspect is especially important regarding security issues. However, with the presented modular approach, parts of the workflow can be easily replaced with more sophisticated online speech recognition services. Due to the possibility of intent and variable detection the speech control allows the execution of more programs and a variation of these (e.g. different lengths of driving straight ahead, etc.). It still needs to be pointed out that the clients/robots need to be independently programmed; therefore, the user needs to be able to program the desired robot. In sum, the proof-of-concept works and there are still options left to improve it.

References

[1] T. Herold and T. Werkmeister, "Practical Applications of Multimedia Retrieval," 7 April 2016. [Online]. Available: https://github.com/twerkmeister/iLID/blob/master/Deep%20Audio%20Paper%20Thomas%20Werkmeister%2C%20Tom%20Herold.pdf.. [Accessed 2019 March 9].

[2] CMU Sphinx, "Open Source Speech Recognition Toolkit," 7 Juni 2017. [Online]. Available: https://cmusphinx.github.io. [Accessed 2019 March 9].

[3] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet and J. Dureau, "Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces," 6 Dezember 2018. [Online]. Available: https://arxiv.org/abs/1805.10190. [Accessed 2019 March 9].

[4] eSpeak, "eSpeak text to speech," 1995. [Online]. Available: http://espeak.sourceforge.net. [Accessed 2019 March 9].

[5] Franka Emika, "Panda," 2018. [Online]. Available: https://www.franka.de/panda. [Accessed 2019 March 9].