COBEM 2023 27TH INTERNATIONAL CONGRESS OF MECHANICAL ENGENEERING

ABCM Brazilian Society of Mechanical Sciences and Engineering
27th ABCM International Congress of Mechanical Engineering
December 04-08, 2023, Florianópolis-SC, Brazil

COB-2023-1007
A STUDY OF THE APPLICATION OF AUTOMATIC SPEECH RECOGNITION IN ASSISTIVE ROBOTICS

Pedro Pereira Nunes
Walter de Britto Vidal Filho
University of Brasilia, Brasilia, DF
pedropereiranunes1999@gmail.com
wbritto@unb.br

Abstract. *Automatic speech-to-text technology is widely spread nowadays by performing an important role by supporting visually impaired individuals in their daily tasks. Another important application is in the robotics field to support life for disabled people. In this case a robot could handle tasks for those with motor limitations, particularly in their arms and hands. This research covers the usability of a voice command interface for a robotic arm, analyzing some common words in Portuguese that could be implemented. Additionally, analyze two main types of commands: one that indicates the coordinate where the robotic manipulator should stop and a command that does not indicate it and therefore needs a stop command. To validate the set of commands, false-positives and false-negatives tests were developed. After that the types of commands quality were measured using time and number of commands to solve a task. Using WIT API for Speech Recognition, more than 30 words were examined with an error rate below 10% and about 25 commands were implemented to control movement and the end-effector. It was observed that the manipulation of a robotic arm was easier when the target coordinates were known previously by the computer, in terms of execution time and practicality, considering that stopping commands need the user's full attention. An interface that uses only voice to control the robotic arm can be integrated, especially if the target coordinates are known.*

Keywords: *Automatic Speech Recognition, Robotic Arm, WIT, disabled people, command interface.*

1. INTRODUCTION

A high rate of technological development has emerged in the world since the beginning of the last century, affecting all scopes of society, to provide an improvement in the human quality of life (Moon *et al.*, 2003). This advance promotes the integration of knowledge from different areas, that generates new technologies or applications for already known techniques. One of the areas that has taken advantage in this development is robotics, integrating new techniques of machine building, robot control and robotic command interfaces. Therefore, different interfaces for operating robots had been developed as Brain Computer Interface (McFarland and Wolpaw, 2008), Gesture Based (Waldherr *et al.*, 2000) and Voice Based (He *et al.*, 2021).

These new interfaces approaches can be deployed in many areas of our society with focus on different aspects of life. Robotic technology is commonly used in industry, which is usually a structured place with predictable scenarios, but developing helpful robots in mankind's daily life is still a challenge because of its unpredictability (Granata *et al.*, 2010), (He *et al.*, 2021). Furthermore, there is an intent to use robots to support people that often struggle in common tasks, such as to eat something, to move an object or even themselves, like individuals that suffer from paraplegia. However, there is a doubt about which interface would be of benefit for motor disabled people. Most of us commonly interact with society using voice and sounds (Granata *et al.*, 2010), as it is a natural and intuitive way of interaction. Hence, a voice-based interface of command could be a good candidate for an assisting robot.

There has been an increase of commercial speech-to-text interfaces. It can come in the form of hardware and also as Application Programming Interface (API) in which a company or individual develops an interface and users can use its programming. There are several brands across the market, and some have already been passed in a benchmarking test, as published by Lima *et al.* (2020) and Filippidou and Moussiades (2020). Both papers measured the quality of Google, IBM and WIT APIs using a Word Error Rate (WER) measure, in which the quality is obtained by a formula considering missed, added, and switched words in the process. It was noted that WIT API has a better speech-to-text quality when working in Portuguese (Lima *et al.*, 2020) and therefore it was chosen for this paper because, besides its quality, it also provides an AI that facilitates the human-computer interface.

Even with a voice interface, there are still various ways of commanding a robot, such as telling it to move along a path to do a task, go to a specific location or controlling each joint separately. Therefore, this paper brings a comparative

between different kinds of commands by analyzing their efficiency in terms of execution time and number of commands. These result, and specially the reason behind it, could provide a sense of how to build a interface to a person with disabilities, since ease and practicality are necessary. For this purpose, some algorithms paths were implemented in order to understand deeper which would be the easiest to be used by a person with disability. Moreover, this paper also wants to provide some data about WIT API, such as word comprehension, false negative rate and mistaken execution. This information is useful as the technology is built to be deployed in daily life and therefore can not risk life by any means.

2. BACKGROUND KNOWLEDGE

2.1 WIT API

WIT API, that can be accessed on Inc. (2023), is an API developed by Meta Platforms Inc, and from now on will be called just by “WIT”, being a framework that allows voice recognition in more than 20 languages, including Brazilian Portuguese. Moreover, its voice recognition is built on the concept of natural language recognition, using an Artificial Intelligence (AI) the software tries to attribute a meaning for the received voice command. WIT also supports a range of programming languages, such as Python, JavaScript, Ruby and others. One of its main functionalities is creating a chat bot for quick messaging, therefore it accepts text and voice messages. In such a way, it would even be possible to try to talk with the robot. Unfortunately, as the common usage of WIT is for chat bot creation, it does not have a continuous speech recognition feature.

Wit is a web server-based API that works trying to comprehend the inputs given by users, thus the first thing to do is training the AI to help the comprehension. Despite some other features available, there is an Understanding tab, illustrated in Fig. 1, that enables insertion of the application’s phrases, called Utterances. Not informed sentences are analyzed according to the similarity with the other registered Utterances.

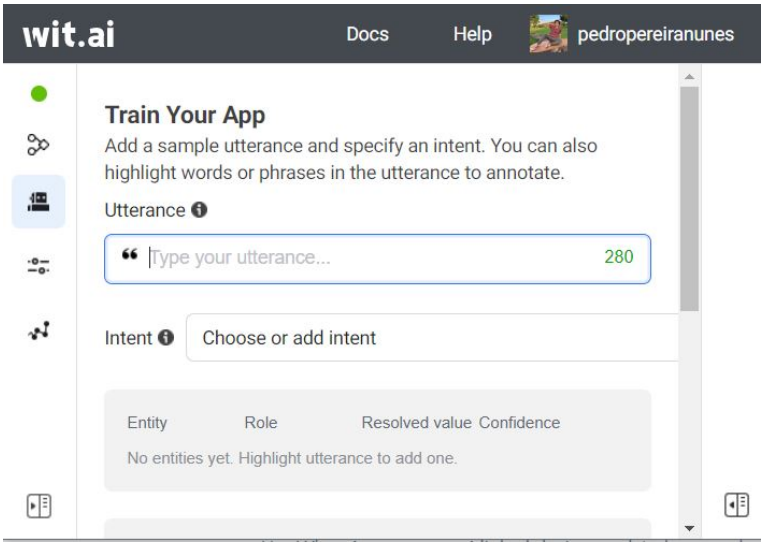

Figure 1. Wit API - Understanding tab

As the API tries to relate meaning for the application Utterance, it is possible to give an intention to a phrase, named Intent. This feature helps in case of existence of different commands that triggers different algorithms of response, in such a way that the developer does not need to analyze the phrase to know its purpose, the API does it. A feasible example in this paper is using the feature for distinguishing commands with known final coordinates and commands in which the robot needs a stop command.

The last basic feature of WIT API is the possibility to discriminate keywords in a phrase, called Entity. The applicability of Entities is upon receiving commands that only vary a little one another. In such a way, the software can register the Entities and pass for the developer the keyword of a given Intent as an argument. In cases when the interface has the commands for picking things, an Entity definition could differentiate the objects available for picking previously.

From now on, this study will refer to Intent as "type of command" and Entity as "keyword", in order to simplify the understanding of the concept.

2.2 Robotic manipulator specifications

Regarding a handicapped person with motor issues, the robot arm thought for the developed interface is a stationary robot that could be linked in a wheelchair or in a workspace environment. Therefore, as it was available for this task, all

tests and development were made in a manipulator in SCARA configuration (Fig. 2). In summary, the robot possesses three joints, in which the first two are rotation joints while the last one is a linear motion joint. Furthermore, this configuration is marked by the axis of movement parallel to its three joints which are vertical in relation to the ground. Figure 3 illustrates the links of this configuration, which portrays the abstract design of the robotic manipulator used.

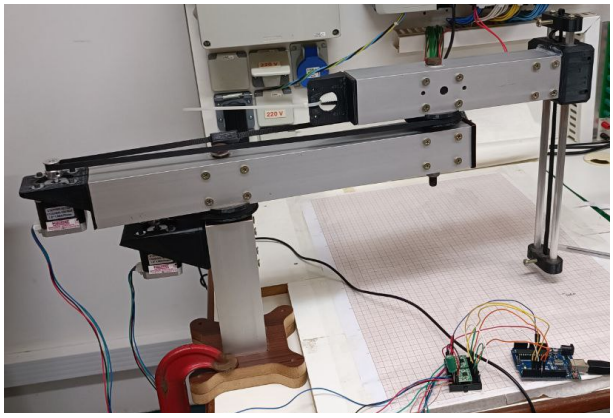

Figure 2. SCARA robot used

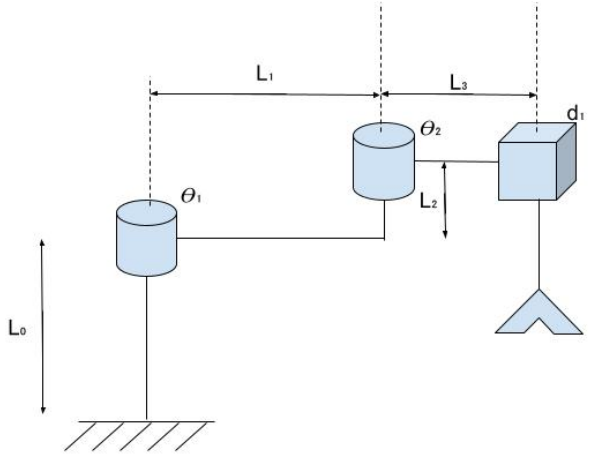

Figure 3. SCARA abstract design

Following the schematic representation, Table 1 indicates the dimensions of each connection. The used nomenclature references the joint moved by θ_1, as A_1, and the joint moved by θ_2, as A_1. More over, The Denavit–Hartenberg (DH) parameters of the robot are shown in Tab. 2.

Table 1. Dimension of robot links

Link	L_0	L_1	L_2	L_3
Length	240	248	70	170

Table 2. The DH parameters of the robot

i	a_i	α_i	d_i	θ_i	**Range**
1	L_1	0	L_0	θ_1	-180º $\sim$ 180º
2	L_3	180	L_2	θ_2	-160º $\sim$ 160º
3	0	0	d_1	0	0 $\sim$ 250mm

Furthermore, the robotic manipulator has its motors controlled by the microcontroller Arduino UNO, which is developed in Arduino platform, using essentially C/C++ language.

3. INTERFACE IMPLEMENTATION DETAILS

This section illustrates details about the developed interface. Section 3.1 shows the architecture perspective and section 3.2 depicts the used algorithms in implementation.

3.1 Interface Architecture

Five main modules compose the developed interface: Main, Text Information Extraction, Motion Planning, Computer-Robot Communication and Arduino, as depicted in Fig. 4. The details of each module is as follows:

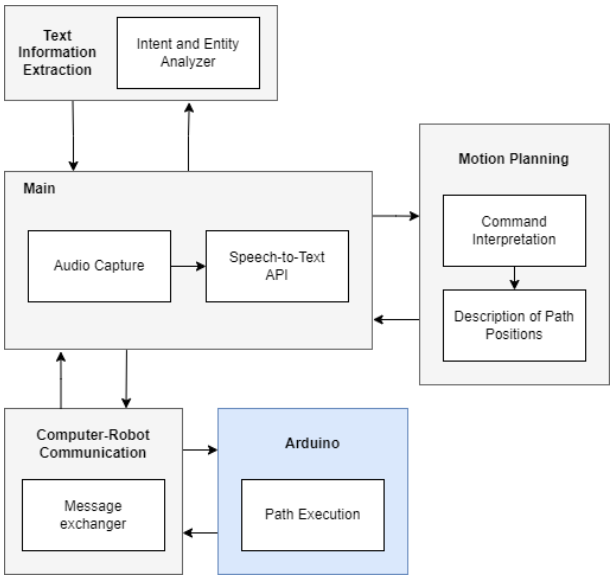

Figure 4. Interface Architecture Model

1. Main: it is the primary module. It functions as a central for the others modules. This module gives an interface to the user and therefore is responsible for capturing the audio that would be recognized. Once recorded, it sends it to WIT for the Speech-to-Text process and receives the text response. Since this module was designed to be a central module, it is also responsible for exchanging information with the other modules and it comprises the algorithm in which the system operates.

2. Text Information Extraction: it is responsible for the next step in the process, obtaining useful information from the text received from WIT. As described in section 2.1, WIT response can be easily analyzed by determining an Intent and Entities for the given command. This study used this characteristic in order to provide an easier interface to develop. Hence, the current module is responsible to extract Intent and Entity information from the WIT response.

3. Motion Planning: it is in charge of the robot arm motion calculus. After the previous module ended its role for the current command, this module determines the strategy of calculus that needs to be used to achieve the command's desired output by the combination of type of command and keywords received. Then, with the forward and inverse kinematics equation, path positions are calculated. The robot uses stepper motors to move, moreover this module is also responsible for converting each path position in steps for the motors.

4. Computer-Robot Communication: the fourth module of this Interface, comprises the exchanged messages between the computer and the robot. The first implementation of this study was done in a computer connected via serial communication with the robot Arduino. Consequently, this module covers the message format that are exchanged, it builds the message using the path positions obtained previously and, finally, sends and receives all messages from the robot microcontroller.

5. Arduino: it is the last component from Interface architecture. It is a module developed outside the main computer, being programmed in the microcontroller. Its principal objective is to execute the path positions described in the received message by applying the number of calculated steps in the Motion Planning module. Since some commands can receive a stop message, the program is responsible to count how many steps were applied before the ending signal and return those values to the main computer.

3.2 Interface implementation details

This section focus on clarifying details about the interface implementation. Here, the programmed commands are illustrated as well as a description of the algorithm that each command triggers for the robot arm.

Differently from other Speech-to-Text APIs, WIT permits its users to determine some of the phrases that the application will receive. Moreover, it is possible to specify different types of commands expressed by each phrase and already select which part of the sentence is a keyword. With this feature, the developer can spend less time processing the commands inputs and discovering their meaning.

This study defined a set of five types of commands, each one of them uses a different strategy for path calculation. All types of commands have a specific WIT intent and one or more keywords. Among those five types, two define commands in which the final position are known, two types need an stop signal when executed and the last one is dedicated to the end effector. The set of implemented commands in this study has only the purpose to move the end-effector around the tridimensional space. Hence, it was not developed a command that would complete an enormous task, such as drawing something, or moving an object by using only one command. The types of commands and the description of theirs algorithm is as follows:

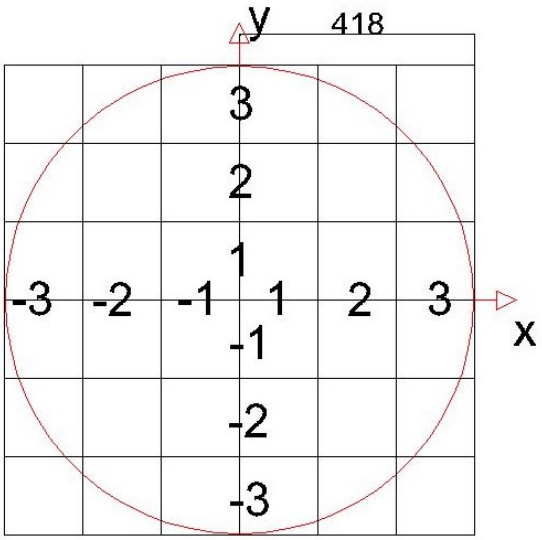

Figure 5. Linear Location quadrant numbers

1. Linear_Location: the first set of implemented commands is composed by those in which a specific coordinate is targeted. Its goal is to bring the end effector close to a determined place in the XYZ space. These commands consider that the robot's workspace is divided in 36 quadrants in the XY plane and in 2 heights in the Z axis. Each command sends the robot's end effector to the quadrant center.

   To simplify, the movement along the XYZ space is limited and the robot arm only travels along XY plane or Z axis. Therefore, when moving in the XY plane, the algorithm calculates points in a straight line between the current position and the desired one. The robot then tries to pass through every calculated point. An exception was made in cases where the current position and the desired one crosses near the robot base. In this situation, the robot makes an elliptical route by only activating both motors with the same velocity.

   Moving through the XY plane requires a command that gives both coordinates, therefore this type requires two keywords in the command. The region were labeled according to numbers, in both directions, as depicted in Fig. 5. The value grows in magnitude as it moves further from the robot's base and are positive when moving in the positive X or Y axis and negative otherwise. In the command, the first number represents the X axis while the second number represents the Y axis. Therefore, as it is seen in Fig. 5, the robot's workspace was divided into 36 quadrants that can be achieved by the current type of command.

   The word "Area" is used to identify this type of command, its structure is: "Area" + Number X + Number Y. In case the movement is on the Z axis, the two heights determined are differentiated as a "Superior" height or an "Inferior" height, therefore the command takes the following form: "Area Superior" or "Area Inferior". This category has 38 defined commands.

2. Radial_Location: has a similar purpose than the previous type of command. However, it treats the problem in a

simplified way by using only one keyword to determine the target position. To achieve this, the θ_2 angle is fixed and consequently only A_1 rotates.

Since only one motor acts, the robot follows a circular path, from where the type name was derived. These commands divides the XY plane on eight slices, similar to pizza slices, and the algorithm moves the robot to the center line that passes through the slice, beginning from the robot base. Differently from Linear_Location, this type of command does not have a predetermined X and Y coordinate, only a fixed angle between base and end-effector. Hence, in every command, the target coordinate needs to be calculated again.

These commands follow the structure: "Arco" + One of the eight main cardinal coordinates in Portuguese. The word "Arco" in Portuguese could be translated as "arc" and the eight main cardinal coordinates is a reference for the words north, east, southeast, etc.

3. Linear_Movement: In the previous two types of commands, the computer assumed that the robot goes to a predetermined final position given by the command. However, the current type was developed so the user could determine a route for the robot to travel by and could also determine a stopping coordinate during execution. To obtain it, the assumption made by the algorithm is that this kind of command determines a trajectory to be followed until a physical limitation, as the robot joins are already fully extended or retracted. And meanwhile the trajectory is being traveled, the user can stop the movement.

   The movement is called linear because these commands algorithm considers the path as a straight line. Therefore, there are basically three kinds of straight lines that could be traveled on: lines parallel to the X or Y axis and the line that passes through the robot base (XY origin) coordinate and end-effector coordinate in the moment the command is said. These travel lines are limited by two possible virtual target positions, the outside range of the robot, when θ_2 is equal to zero, and the inner range of the robot, when θ_2 is equal to 160º. In both cases, if this position is achieved, the robot could not go further while still following the straight line assumption. Hence, the algorithm calculates the limiting coordinate and uses it to virtually achieve a target position. It is also possible to move along the Z axis and, in this case, the algorithm considers the maximum and minimum range of the linear joint as the target position. Figure 6 exemplifies in which direction the commands send the end-effector in the XY plane.

   This type of command uses the word "Garra", translated as "Craw", referring to the robot craw. Furthermore, the keywords for movement in the XY plane are: “direita, esquerda, frente, trás, avançar and recuar” (in English, it would be: right, left, forward, backward, advance and retreat). Right and left define the parallel line to the Y axis, forward and backward define the parallel line to the X axis and advance and retreat define the line between base and end effector. For movements along the Z axis, the keywords are: "subir and descer" (move up and move down, respectively in English). The command structure is a two-word sentence composed of: "Garra" + keyword. Additional eight commands were implemented with this algorithm.

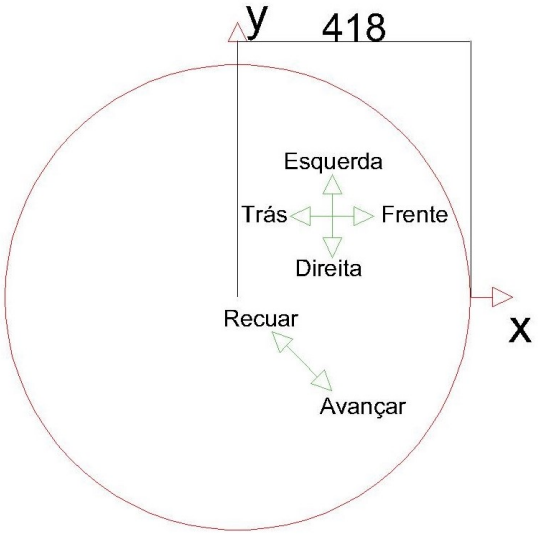

Figure 6. Linear_Movement commands direction

4. Radial_Movement: in the last group of commands for moving the end effector around the space. Similarly to Radial_Location, it intents to simplify the previous one, using circular paths. In this category, the command triggers only joint A_1 or A_2 to move the end-effector along the XY plane.

The proposed movement could be in the two directions for each motor, a clockwise rotation or a counter clockwise rotation. Two parameters are necessary to define it, the motor and the direction. Therefore, it uses the following structure: "Junta" + Joint number + Direction. Junta is the Portuguese word for "Joint" and the direction keyword is "Clockwise" or "Counter Clockwise" said in Portuguese.

5. End_Effector_Control: it is used to switch the craw states. The craw used has only one degree of freedom, therefore when used, it can only close and open in a basic transition.

   The developed commands for this type express this idea by simply being structured as: "Efetuador" + "Abrir/Fechar", in which "efetuador" means "effector" and "abrir/fechar" means "open/close".

4. EXPERIMENTS AND DISCUSSION

In this section, the interface is analyzed by testing some attributes such as its recognition success rate for words and commands and also the time needed to complete a task. Moreover, the time that WIT needs to process the voice input was measured.

4.1 Recognition Delay

Considering that the interface is desired to be used in daily life applications of disabled people, it is important to understand aspects that affect the Quality of Service (QoS), as the waiting time needed to the robot starts acting, the delay. Therefore, the first carried out test consists of measuring the response time of the platform WIT when a voice input is sent. Thus, the main goal here is to obtain the necessary time that the platform needs to analyze the audio file, transcribe it to text and return it.

To perform this task, 150 time samples were collected. The samples were obtained by sending to the interface a command from all the five groups of commands that compose the interface. The lengthier commands from the developed interface were selected.

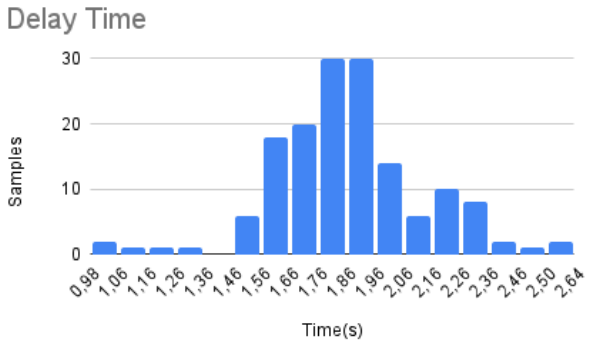

Figure 7. Voice Recognition Delay Time

It can be seen from Fig. 7 that it usually takes under than two seconds for the platform to transcribe the sent audio, despite the time needed to speak. The collected data show a mean of 1.86 seconds with a variance of $0.069s^2$, therefore the time samples stay close to the mean.

Notice that the delay time is correlated with the available internet bandwidth. Therefore, if the test is performed in a quality below than 200Mbps, which the test was submitted, the result can worsen. This is important, since the robot could be installed in a wheelchair and lose the QoS if the available internet decays.

4.2 False Negative Rate

The necessity to repeat a command because of misinterpretation of the WIT also affects the QoS. Therefore, the next performed test resolves around the false negative rate concept, measuring the percentile of times in which a said word was incorrectly recognized. 28 samples were collected per word and the obtained percentile shows how much of false negative cases happened.

All words that composed the interface were tested, but only those that showed an error rate above 0% are displayed in Tab. 3. Each word tested was also classified in the three types of errors concerning recognition, as mentioned by Filippidou and Moussiades (2020): Substitution, Deletion and Insertion.

The deletion happens when the word to be recognized is deleted in the process and Tab. 3 shows that it is the most common case of error, however in the words that it occurs, the error has an average of only 9.329%. Furthermore, the substitution case, when there was substitution by another word, consists of 40% of the error with an higher error average

Table 3. False Negative rate

Word	Error Rate(%)	Error Type	Word	Error Rate(%)	Error Type
Área	9,09	S	Sudeste	11,10	D
Um	22,00	S	Sudoeste	8,30	D
Dois	17,4	I	Sul	17,30	D,S
Menos	19,20	S	Superior	3,80	D
Árco	7,10	I	Inferior	9,09	D
Norte	16,67	D	Junta	10,70	I
Noroeste	8,33	D	Garra	6,67	D
Leste	24,00	S	Subir	6,20	S
Oeste	8,70	D	Descer	23,00	S
Recuar	3,33	S			

of 27.6%. Lastly, the insertion, when something else is added to the transcription, is observed in only three cases, having an average error rate of approximately 12%.

During the testing, it was noticed that despite being the most common cause of error for lots of words, a deletion happens only a few times, less than 10%. Hence, it is possibly an error that occurs eventually caused by the speaker's execution as saying too fast or far away from the microphone. It is also important that deletion happens on average less than the others, since it is less treatable, considering that substitution and insertion can be treated in some cases given that there is still a word to treat.

The substitution case indicates that a word is easily comprehended as other words. Therefore, it occurs with words that is phonetically similar to others, even in other languages as the word "Leste" from Portuguese is equally spoken as "Last" in English. Those cases could be treated if those similar words are not used in the interface, by also adding them to the list of commands. However, it is necessary to know every similar word that can appear and add them.

In parallel with substitution, the insertion error showed itself usually as a comma added to the transcription. It can also be easily treated by removing the commas from every text received.

4.3 Mistaken Execution Rate

Meanwhile the previous test tried only a word at turn, the mistaken execution rate test is proposed to analyze if the commands will be executed by the interface in case of miss-pronunciation of some commands. This test is needed since WIT has an AI that label the transcription with an Type of Command and Keyword. Hence, if any of them are wrongly returned, the interface should execute an unpredictable command.

For this test, it was selected 8 commands that do not compose the interface, but with a similar sound to commands that do compose. The sound similarity was obtained by substituting words. In half of the tests the substituted word was a keyword. And the rate measured reflects if the wrongly said commands triggers the software to do some action. Every command was said 15 times.

Table 4. Mistaken Execution Rate

Input command	Execution rate(%)
Área menus um dois	100,00
Pária um dois	85,70
Arco Forte	0,00
Marco Nordeste	0,00
Inunda um horário	100,00
Junta um armário	0,00
Garra relutar	0,00
Marra avançar	100,00

Differently from the false negative test, the results shown in Tab. 4 are more at the limit of happening all time or never happening. This is because the API always analyzes the wrong input command in the same way. It was noticed that when the command has a miss-pronunciation in a keyword, the software often does not execute, even if a type of command is perceived. This effect happens since the keyword wrongly recognized is not know by our interface. In other hand, if the first word of the command is changed, that was used to identify the type of command, then it often is executed, probably

because WIT tries to give a meaning for the command that has a known keyword and format.

Sometimes, commands will be miss-pronounced, and it can be good for the interface to execute them. However, there are cases that need to be treated as the first command of Tab. 4. It was a keyword composed of two words "Menos um" ("Minus one", in English). In this case, the word “Minus” was switched by a very similar word, consequently the API stopped considering the wrong word as part of the command. Hence, instead of executing "Area minus one two", it was executed "Area one two", which is completely wrong. Those unacceptable cases could be treated by not trusting entirely in the type of command and keyword response, but also matching the result with that which would be expected for that type of command. u

4.4 Difference between types of commands

This paper tried to identify how differently ways of commanding an arm manipulator affects performance. Since the interface developed is voice-based only, it is important to pursue the difference between types of commands. Therefore, the performed test in this section tries to analyze some contrasting characteristics between the commands that go directly to a position in space and those that move along a trajectory and need a stop command. The main characteristics that were obtained is time to do a task, i.e. Execution Time, and how many commands it was needed for executing it, i.e. Number of Commands, since both also could affects QoS.

To perform the current test, the same task needs to be accomplished by the groups of commands. The stipulated task is as follows: go to zone 1 3, from Linear_Location type, move the end effector down, grab an object, move the end effector up, go to zone 3 1, also from Linear_Location type, move down the craw and release the object, composing a total of seven actions needed.

Three groups were specified in order to analyze the distinct ways of interacting by voice. The first group uses the commands from the Linear_Location type, the second uses the Radial_Movement type and the third uses the Linear_Movement type. It is also important to note that all of them also uses the End_Effector_Control type of command. Moreover, each group of command were tested ten times.

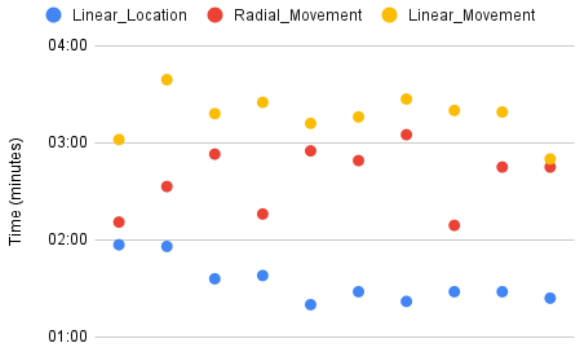

Figure 8. Task execution time

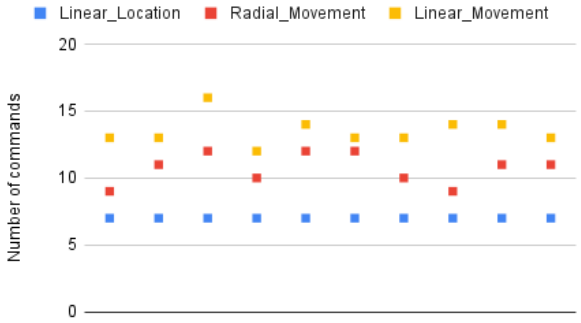

Figure 9. Number of commands

From Fig. 8 and 9, both displayed as scatter plots, is possible to notice that execution time and number of commands needed to execute the task has an hierarchy in which Linear_Location precedes Radial_Movement and at last Linear_Movement. The specified task is composed of seven actions and therefore Linear_Location would only need seven commands to execute it. It was also expected that the others groups have necessity for more commands, but distinctly from Linear_Location, this value oscillates. The oscillation is probably derived from the stop command, since

when the users stops at the wrong position, it is needed more commands to compensate. Moreover, as the workspace from a SCARA robot is a cylinder, Linear_Movement that only travels parallel with the axis and not diagonally needs even more commands.

The increase in the number of commands affects the needed time to execute the task, given that more time to say the commands is required. Therefore, it also makes sense that the execution time is greater for the commands that need a stop command. Therefore, meanwhile Linear_Location type required an average of 1:28 minutes to complete the task, Radial_Movement and Linear_Movement required an average of 2:45 and 3:18 minutes, respectively. Linear_Movement practically doubled the execution needed time to solve a task. The needed time to complete a task also levitates inside the group, once the user starts to get familiarized with the interface, one tends to get faster while using it.

5. CONCLUSION

In this paper, a voice-based interface was developed to control a robotic arm with the aim of supporting people with disabilities. An architectural model for a voice interface was proposed using the WIT, but it is possible to adapt it to any other Speech-to-Text software. The developed interface was tested to measure the quality of voice transcription and the distinct characteristics that different types of commands present.

In this version of the interface, the commands that control the position and path of the robot were implemented and their differences were discussed. It was observed that sending the robot arm directly to a specific location presents better. In this version of the interface, the commands that control the position and path of the robot were implemented and their differences were discussed. It was observed that sending the robot arm directly to a specific location presents better results then conducting it step by step to the desired position. In the first case, it obtained better performance in terms of execution time and number of commands needed to complete the tasks. It was also noticed that the number of commands needed vary for those with a stop signal, since they need a visual confirmation from the user to stop. Consequently, this types of commands can be difficult to use, specially for those with motor limitations in the neck. This problem could be minimized with the integration of other interface, as a graphic interface that shows in which position the end effector is.

The interface commands (words) were also tested to verify the API success rate. Three types of errors were tested: deletion, insertion, and substitution. The main cause of error was the deletion of words, but with a lower error rate than substitution or insertion as discussed in the text. Deletion could not be processed, considering that there is no word to be treated, hence this result of low error rate is important. Substitution errors occur when a word used is very similar to another, therefore it would be recommended to choose distinct words to compose the command. Insertion errors occurred frequently in cases where the word was followed by a comma, which can be easily processed and removed.

In order to improve the interface in future works, it is needed to implement commands to complete a whole task such as moving an object through the space. It would be also valuable to measure the QoS parameter in real world, in such a way that users would describe if the delay time measured is good enough to use.

6. REFERENCES

Filippidou, F. and Moussiades, L., 2020. "α benchmarking of ibm, google and wit automatic speech recognition systems". In I. Maglogiannis, L. Iliadis and E. Pimenidis, eds., *Artificial Intelligence Applications and Innovations*. Springer International Publishing, Cham, pp. 73–82. ISBN 978-3-030-49161-1.

Granata, C., Chetouani, M., Tapus, A., Bidaud, P. and Dupourqué, V., 2010. "Voice and graphical -based interfaces for interaction with a robot dedicated to elderly and people with cognitive disorders". 19th IEEE International Symposium on Robot and Human Interactive Communication, Principe di Piemonte - Viareggio, Italy.

He, Y., Deng, Z. and Zhang, J., 2021. "Design and voice-based control of a nasal endoscopic surgical robot". *CAAI Transactions on Intelligence Technology*, Vol. 6, p. 123–131.

Inc., M.P., 2023. "Wit api". `http://Wit.ai`. Accessed: 2023-09-11.

Lima, M.N.S.M., Coelho, B. and Takigawa, F., 2020. "Ferramentas e recursos disponíveis para reconhecimento de fala em português brasileiro". Computer on the Beach, Santa Catarina, Brazil.

McFarland, D.J. and Wolpaw, J.R., 2008. "Brain-computer interface operation of robotic and prosthetic devices". *Computer*, Vol. 41, pp. 52–56.

Moon, I., Lee, M., Ryu, J. and Mun, M., 2003. "Intelligent robotic wheelchair with emg-, gesture-, and voice-based interfaces". Intl. Conference on .me1 agent Robots an0 Systems, Las Vegas. Nevada.

Waldherr, S., Romero, R. and Thrun, S., 2000. "A gesture based interface for human-robot interaction". *Autonomous Robots*, Vol. 9, p. 151–173.

7. RESPONSIBILITY NOTICE

The authors are solely responsible for the printed material included in this paper.