A Software Testbed for Assessing Human-Robot Verbal Interaction

by

Hassene Bouraoui

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Sciences
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2010

AUTHOR'S DECLARATION

Abstract

Verbal interaction provides a natural and social-style interaction mode by which robots can communicate with general public who is likely unknowledgeable in robotics. This interaction mechanism is also very important for a broad range of users such as hands/eyes-busy users, motor-impaired users, users with vision impairment and users working in hostile environments. Verbal interaction is very popular in robotics especially in personal assistive robots, which are used to help elderly people and in entertainment robots. Several research endeavors have been assigned to endow the robots with verbal interaction as a high-level faculty. However, the language usages of many of them were simple and may not be considered as full speech dialogue systems providing natural language understanding. In this thesis, we investigate a testbed platform that can be deployed to enable human-robot verbal interaction. The proposed approach encompasses a design pattern-based user interface and a user-independent automatic speech recognizer with a modified grammar module in the context of human-robot interaction. The user interface is used to simulate robots response toward multiple users' voice commands. The performance of the proposed testbed has been evaluated quantitatively using a set of evaluation metrics such as word correct rate, recognition time and success and false action rates. The conducted experiments show the promising features of the system. The results obtained could be refined even further by training the system for more voice commands and the whole system could be ported to real robotic platforms such as Peoplebot to endow it with natural language understanding.

Acknowledgements

First, I would like to express the deepest appreciation to my supervisor, Professor Fakhreddine Karray, who has the attitude and the substance of a perfect supervisor: he continually and convincingly conveyed a spirit of adventure in regard to research and scholarship. I would like also to thank him for his endless support and guidance on this research work.

I am also very thankful to my co supervisor Dr.Alaa Khamis for his valuable guidance and generous help throughout my research work. He has been a great mentor throughout this work, I learned from him a lot.

Many thanks are also due to my thesis readers Dr. Slim Boumaiza and Dr. Mohamed Dabbagh for taking the time and assessing my work within a short time frame. Special thanks are due as well for Dr. Jiping for all the fruitful discussions and feedback. The author acknowledges Vestec Inc. for providing valuable resources used in this thesis and for its courtesy to use its labs to carry out some of the reported experiments.

I would also like to thank the group in the PAMI lab for helping me out to port some of the functionalities of the system to the Peoplebot robot platform.

To my friends I would like to express my sincere appreciation for their invaluable support and suggestions of this dissertation.

Last but not least my lovely people. Special grateful to my mother and father who give me all the love and support I need throughout my life and work so hard during their lives to make me a better person. May Allah reward them in this world and in the hereafter. My warm thanks to my lovely sister Hasna for her love and support. During the last two years I was exposed to a lot of pressure and challenges and without her support and wisdom I would not make it through. My wholehearted thanks go to the persons I love for their encouragement, caring and understanding during this period.

Dedication

To my parents, sister and family

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

As robots become more integrated into our society, it is increasingly important to develop natural or social-style interaction mechanisms by which these robots can communicate with general public who is likely unknowledgeable in robotics. Unlike industrial and laboratory robots, which perform their tasks in highly structured environments populated by trained professionals, social robots are meant to function as part of the everyday life of a more generalized public [1]. Building a social robot requires contributions from different fields such as human-computer interaction, artificial intelligence, robotics, computer vision, natural language understanding, and social science (psychology, cognitive science, anthropology, and human factors). One of the fundamental requirements of social robots is the ability to function more naturally as partners for the human not just as mere tools. These robots need to interact with human (and perhaps with each other) through similar ways by which humans interact with each other. To achieve this goal, novel interaction methods must be developed in order to allow humans to interact more naturally with the robots by employing human-like social norms and interaction modalities. Integrating multimodality and adaptivity can help in developing social human-robot interfaces [2]. Multimodality allows humans to move seamlessly between different modes of interaction, from visual to voice to touch, according to changes in context or human preference. Multimodality has implications for human-robot interface usability and accessibility. Last decade has witnessed intense research activities in human-robot social interaction. This research activity has borne some fruit in tackling some of the challenging problems of human-robot social interaction that are still open. These research problems include, but are not limited to, robot autonomy, verbal and non-verbal (embodied) interaction, human and domain modeling, learning from communication and learning to communicate, context awareness and intention recognition, compliance, safety and compatibility, human-robot cooperation and interacting with groups of people. This thesis tackles one of these problems –verbal interaction, which provides natural communication medium between the robot and the human.

1.1 Motivation

Human-robot interfaces facilitate communication, assist on the exchange of information, process commands and controls and perform several additional interactions. Spoken natural language is more user-friendly mean of interacting with a mobile robot. From the human perspective point, this kind of interaction is easier since it does not urge humans to learn additional interactions. Humans can rely on natural ways of communications instead.

Human-robot verbal interaction varies from understanding simple commands to extracting all the information in the speech signal such words, meanings and emotional condition of the user. To develop an interface with natural language understanding ability, many issues must be taken into account such as dealing with the ungrammatical nature of many spoken utterances, the detection of errors in speech recognition and interpretation, and the design of intelligent clarification dialogues. The realization of the potential of verbal interaction also requires the solution of other challenging problems such as dealing with frequently and varying background noise, word echo and repetition, and also with different and background sound sources overlapping the speech [3]**.** Most state-of-the-art of the human-robot verbal interaction based systems relies on using simple speech commands and may not be considered as full speech dialogue systems. Having these concepts in mind and addressing human-robot verbal issues, speech dialogue systems should be well developed to be robust enough for use in effective human-robot verbal interaction. Robots should operate more complex commands and be able to go beyond recognizing simple stand-alone commands. The system we are proposing here consists of providing an enhanced speech dialogue framework and can be used as a testbed to study human-robot verbal interaction.

1.2 Objectives

Our principal goal in this research work is to implement a testbed for enabling human-robot verbal interaction and to provide an enhanced speech dialogue framework. To achieve this, the following integrated tasks should be completed:

- Build a java application module that contains the design pattern-based interface employed for simulating the actions of the robots
- Modify on the Julius Speech Recognizer engine module in order to apply and create a user-independent automatic speech recognizer with a modified grammar module in the context of human-robot interaction.
- Evaluate the system output results using multiple predefined metrics, compare these results with existing verbal based systems and check if they are conclusive enough to adopt this approach.

1.3 Contributions

To achieve the stated tasks, we have proposed in this work to integrate in a novel way a more flexible speech recognizer module in order to understand more complex commands based on natural language understanding. Along the way, we have analyzed the results of recognized voice commands to find ways how to improve our system. The Julius speech engine system was altered to satisfy better speech understanding accuracy. In order to check the effectiveness of the interaction interface, a Java user interface was created. The user interface depicts the interaction of the robots with multiple human voice commands. The two created models were used to show how robust the system is using the verbal natural user interface.

1.4 Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 reviews the state of the art of the human-robot interaction and human-robot verbal interaction approaches used in today's interaction systems. Then chapter 3 gives an overview of the architecture of the system and describes the two modules used to implement our approach, the design pattern-based user interface and the user-independent automatic speech recognizer with a modified grammar module. This chapter describes as well the enhanced speech dialogue framework used to enable the human-robot verbal interaction. Chapter 4 introduces the obtained results from computing the different metrics against multiple speech commands variations and then discusses the performance of the system in terms of the proposed testbed for assessing human-robot verbal interaction. Chapter 5 summarizes the contributions of this thesis and introduces the focus of future research. Finally, Appendix A provides a user guide that describes how to use the developed system.

Chapter 2

Speech Recognition Techniques: A Review

This chapter highlights some research work in the speech recognition techniques. It also describes the human-robot interaction and reviews the state-of-the-art of what has been achieved in human-robot verbal interaction.

2.1 Use of soft computing and intelligent systems in speech recognition

Speech recognition is also called automatic speech recognition. Broadly speaking, speech recognition represents the process by which a computer recognizes what a person says. Speech recognition is used in telephony applications. The computer or the system can recognize what the human said when it asks you for instance to give the name of the person you are calling in a company. Speech recognition systems are continuously affordable and their number is increasing due to the exponential decrease in computing cost added to the exponential increase in computing power. People speaking to one other can recognize words spoken as well as the meaning behind these words. Computers are able to recognize individual words and phrases but still lack the understanding of the real meaning of the words as humans do. Speech recognition is divided into speaker independent and speaker dependent one. For speaker dependent software there is a training of the system to the unique characteristics of a single person's voice. On the other hand, speaker independent software is proposed to recognize anyone's voice. This section falls into 3 parts. In the first subsection, soft computing and intelligent systems techniques for speech recognition are described and analyzed. In this subsection, the use of fuzzy logic in speech recognition is first underlined. Then, the application of neural networks in speech recognition is presented and finally some other used techniques are discussed. In the second subsection a summary of the discussed techniques is illustrated and suggested propositions are shown. Finally, the last subsection summarizes the use of soft computing and intelligent systems in speech recognition.

2.1.1 Speech Recognition Techniques

- **Use of fuzzy in speech recognition**

The PROSBER (PROSody Based Emotion Recognition) [4] is a fuzzy rule based approach used in the robot MEXI to recognize emotions from the prosody of natural speech. MEXI was used to recognize emotions from natural speech input and produce natural speech output based on emotions. The emotion recognition was tested on a speaker-dependent mode as well as on a speaker-independent mode. In order to recognize emotion from natural speech, PROSBER accepts single sentences as input and classifies them into five categories: happiness, sadness, anger, fear and neutral. It automatically creates the fuzzy models for emotion recognition. Working modes are divided into training and recognition mode. In training, examples with well-known emotion values are used to generate the fuzzy models for the individual emotions. These models are then used in the emotion recognition method to classify unidentified audio data. Both training and recognition mode performs similarly in four steps as illustrated in Figure 2.1. However, the training mode differs from the recognition mode by employing the fuzzy model generation instead of the fuzzy classification in the fourth step. Recognition working mode utilizes fuzzy classification as follows. The five produced fuzzy rule systems are used to match the actual speech sample with the appropriate emotion degree from the five emotions degrees. Therefore, each fuzzy emotion model picks up the related features selected for the respective emotion during the training.

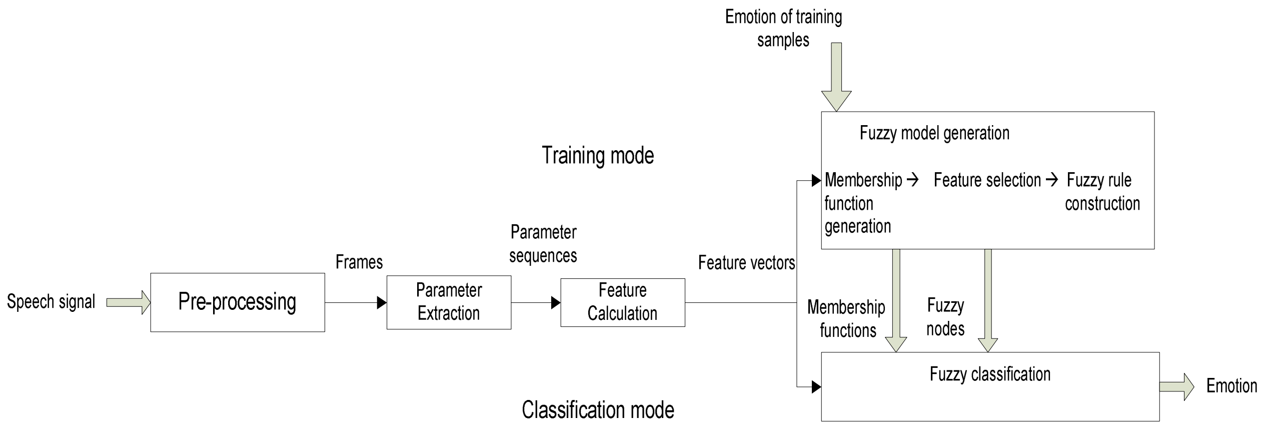


Figure 2.1 - Architecture of PROSBER [4]

The PROSBER system chooses automatically six most important features from a set of about twenty analyzed speech features to create a rule system for each emotion to recognize. Hence, a real-time dialogue is carried out between MEXI and its human counterpart since the recognition complexity is really reduced. Results showed that recognition rates in speaker-dependent mode have attained 84% and 60 % in the speaker-independent mode. Psychologists have reported that the recognition rate is similar to humans in the speaker-independent mode [4] [5].

Karray et al. [6] introduced a fuzzy logic based system for natural language understanding. This system differs from the PROSBER by parsing speech recognition results into conceptual structures in a robust manner, and therefore is able to handle noise caused by speech recognition errors. It obtains conceptual knowledge from corpus data and classifies such knowledge into fuzzy logic inference rules. Referring to some functionalists, linguistic categories are known as gradient and fuzzy as well [7]. Karray et al. have included the idea of fuzzy grammar for natural language understanding. This idea consists on that any match between a word and a category or assignment of a relation between words can be analyzed as gradient. The mode adapted was similar to fuzzy control models when a set of fuzzy conditions activate an evaluation function. The evaluation then passes a threshold to make a crisp decision. In their model of language understanding, the conditions represent context words, which are used to assess the possibility of assigning certain semantic functions to some semantic words.

The used fuzzy logic system achieves two main tasks. Given the context supplied by the recognized sentence, the system, first, evaluates whether a recognized word is semantically suitable. Then, the second task consists of assigning affordable semantic functions to the words that are already successfully evaluated. The system uses important information which is the contextual word rate. This information is obtained from training corpus and hence transferred into fuzzy rules. The used fuzzy semantic logic is composed of three-tuple L (V, F, and R), where V is a set of linguistic vocabulary, F is a set of vocabulary features and R is a set of inference rules.

A member in V is called a word. A member in F is a symbolic feature. The system utilizes the feature types {Semantic-Function, Distance, and Weight}. "*The semantic functions are defined domain specifically. For example, in the air travel domain, words for cities can have the functions "loc-to" (toward location), "loc-fr" (from location), "loc-tr" (through location), etc. Distance is an integer indicating the distance between this word and a word to which the semantic function may be assigned. Weight is a decimal reflecting the relative importance of this word in assigning the semantic function to another word*" [6].

The set R is composed of evaluation rule and assignment rule. Some notations are used to generate the evaluation and assignments rules. For instance, for each word X, if its feature vector encloses a semantic function F, this is symbolized by X*F. On the other hand, for each word Y, if it can possibly take the semantic function F, it is symbolized by Y, if it can possibly take the semantic function F, it is symbolized by Y#F. Symbol '^' means logic And. Symbol '¬' refers to a binary complement. Rules are expressed as follows:

E-rule:

$$Y\#F \wedge X*F \wedge X\ d.w.\ Y \rightarrow F^{Y} + E(F,d,w) \tag{2.1}$$

E-rule:

$$Y\#F \wedge \neg X*F \wedge X\ d.w.\ Y \rightarrow F^{Y} - E(F,d,w) \tag{2.2}$$

Features such as d and w are obtained from training data.

Karray et al. [6], have implemented the fuzzy logic-based (Natural Language Understanding) NLU including a knowledge acquisition component, a natural language understanding component and a performance report component. In order to develop a powerful logic system, large quantities of data is needed either from knowledge experts or from automatic knowledge acquisition processes. As a matter of fact, number of actions was developed to extract information for a fuzzy semantic logic. Automatic acquisition of weights, which is included in the knowledge acquisition component of the system, was one of the procedures used. It uses a shallow syntactic parser to parse the training sentences from the (Air Travel Information System) ATIS corpus [8] into three kinds of phrases: verb phrase, noun phrase and prepositional phrase. The parser shows good results when tested for 13,000 sentences contained in the ATIS corpus. In fact, all of them were parsed. One hundred sentences were tested arbitrarily and illustrated above 97% of correctness in parsing sentences into phrases. During the training stage, 160 reference sentences were automatically parsed into phrase structure using the shallow parser. In order to process the training data, the knowledge acquisition component was used. Two lists were generated. The first one contains 165 semantic words as well as semantic functions labeled on them. The second list contains 1200 context words with feature vectors. Context word features, acquired in the training stage, were applied by fuzzy rule inference rules to build inference.

Recognition results concerning the correct rate of the recognized words as well as the accuracy of the whole system using the 160 sentences were calculated by the understanding component and figured out as shown in the table below.

No. of words	Correct Rate	Accuracy
1983	84.31 %	81.76 %

Table 2.1 Recognition result (160sent)

In order to measure flexibility to noise, other fuzzy technique was used for recognizing speech containing noise. A fuzzy classification combined to a Multi-Layer Averaging (MLA) algorithm has been employed in order to recognize spoken words of a single speaker [9]. The major idea of the MLA is that a given sample is matched to a number of samples already analyzed by the system. This matching is therefore used to eliminate non-similar samples. This is done by attributing a penalty to these samples. Hence, the perfect match corresponding to the most similar samples has the least penalty. To eliminate samples, dissimilarity is used. Two series of numbers may not be similar if their averages are not similar. Based on this idea, the MLA attributes iteratively penalty points to samples by comparing their averages. Comparison is made in different layers. The policy used in MLA for eliminating non-similar samples can be considered as fuzzy classifier. In addition to the MLA, the fuzzy classification was used based on fuzzy numbers in order to recognize spoken words. The fuzzy numbers are represented by the averages of the tested samples which were used for calculating difference in the MLA approach. A test sample (Ai) was used and its averages were considered as fuzzy numbers. These fuzzy numbers are sketched and described by a symmetrical trapezoidal membership function.

Afterwards, membership grades of corresponding matching sample (Bi) averages were calculated. Then, the penalty of all pairs of averages is calculated using a specific formula. The fuzzy classifier is working as follows. The membership value is 1 when two averages are sufficiently close. In this case the corresponding penalty is 0. Otherwise, the value gets a linear penalty of max 100. The fuzzy number (Ai) and the penalty given to a corresponding average (Bi) is illustrated is Figure 2.2.

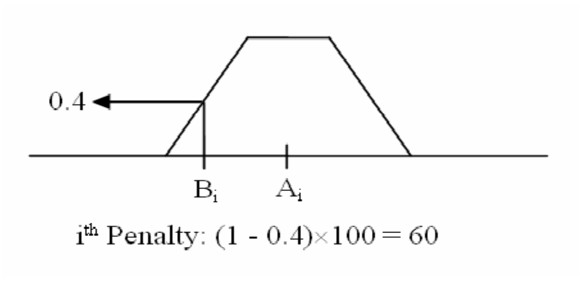
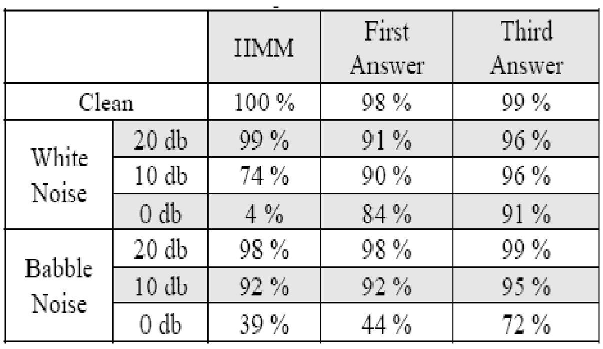


Figure 2.2 - Fuzzy number and penalty calculation [9]

The implemented algorithm was tested and compared to the broadly used speech recognition method, (Hidden Markov Model) HMM. When tested for a single speaker with 100 words, the algorithm outperforms HMM method and shows more flexibility regarding noise. Different kinds of noise were integrated to test data. Hence, two kinds of noises, called Noise and one of the most challenging noise interference for all speech systems called Babble noise, were added to test data and raised again for recognition. Clean data is also used for recognition. The corresponding results are shown in table 2.2 [9]. Referring to results, the HMM algorithm decreases sharply when adding noise. However, the fuzzy based algorithm maintains its effectiveness even with severe noise. The "First answer" denotes first recognized answer is the correct answer. "Third answer" signifies one of the first three answers is the correct answer.



		IIMM	First Answer	Third Answer
Clean		100 %	98 %	99 %
White Noise	20 db	99 %	91 %	96 %
	10 db	74 %	90 %	96 %
	0 db	4 %	84 %	91 %
Babble Noise	20 db	98 %	98 %	99 %
	10 db	92 %	92 %	95 %
	0 db	39 %	44 %	72 %

Table 2.2 Testing results [9]

A novel approach to speech recognition using fuzzy modeling has been also developed by R. Halavati et al. [10]. This approach is based on alteration of speech spectrogram into a linguistic description using random colors and lengths. Then, phonemes are depicted using fuzzy measures. Fuzzy reasoning is also used to perform recognition. Besides the use of fuzzy measures, genetic algorithm is used for optimization and classification of phonemes. The recognition process consists of first computing the belongness of each frame. This is done by calculating the frame's frequency band values through comparing it against its color spectrum. Finally the total belongness is the minimum of all the previously computed values. This first step is demonstrated in Figure 2.3 [10]. A filtration process is then applied for resulting values to decrease the effect of noise and contrast in patterns. The refined belongness measure of each frame is the input of the last step. The matching length is defined as the longest sequence of frames whose belongness are above a specific threshold value. The final result is calculated by multiplying the matching value by the phoneme's belongness value (Figure 2.4) [10].

In addition to the recognition process already applied, a training approach is also used to classify phoneme samples. The training algorithm is supported by using a normal genetic algorithm [11]. In this algorithm, a genome represents the complete recognizer involving color definitions, length definitions and all phonemes descriptions. The training process starts with arbitrary genomes which are sorted iteratively given their fitness.

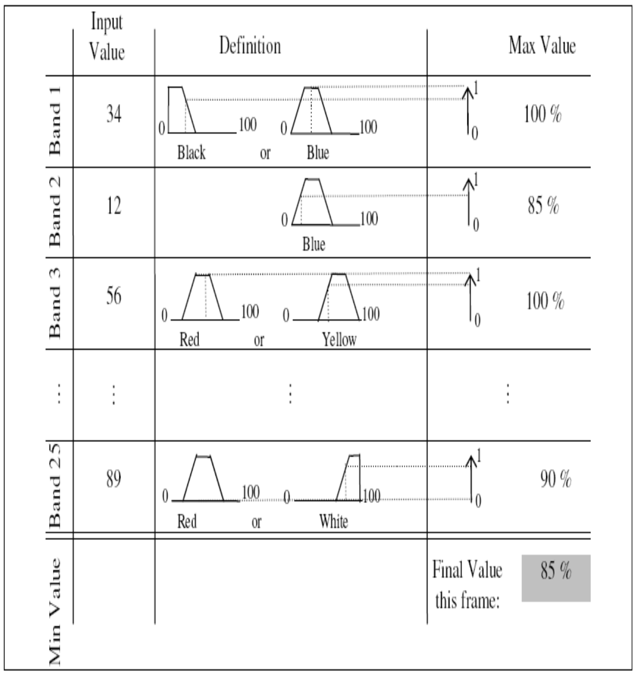
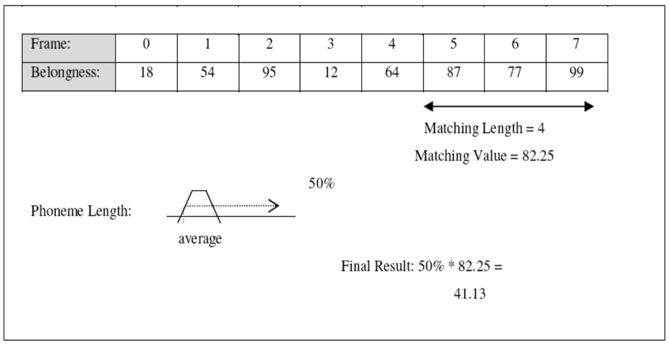


Figure 2.3 - Belongness Computation for one frame [10]



Frame:	0	1	2	3	4	5	6	7
Belongness:	18	54	95	12	64	87	77	99

Matching Length = 4

Matching Value = 82.25

50%

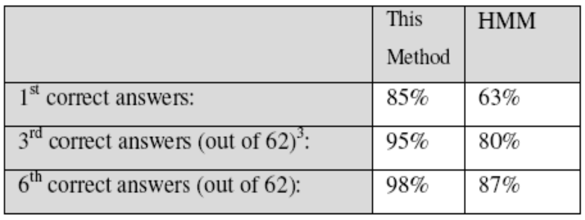
Phoneme Length:

average

Final Result: 50% * 82.25 = 41.13

Figure 2.4 - Computation of final step [10]

Applied to a database with 62 phonemes classes, this algorithm shows good results when tested for a single speaker sample set. This algorithm surpasses the known Hidden Markov Model (HMM) used approach as illustrated in the table below [10].



	This Method	HMM
1st correct answers:	85%	63%
3rd correct answers (out of 62)[3]:	95%	80%
6th correct answers (out of 62):	98%	87%

Table 2.3 Experimental results [10]

- **Use of neural networks in speech recognition**

Speech recognition techniques based on neural networks have been extensively studied in literature. Among the proposed techniques, Majewski and Kacalak present a system for the recognition of commands in natural language [12]. The system utilizes two neural networks to achieve its goal. The first deals with recognizing separated words from text generated by the speech recognition engine. These words are then passed to the command syntax analysis module for processing. Here, the second neural network is used to either recognize the commands or reject them. The module employs a 3-layer Hamming neural network as shown in Figure 2.5 to perform its operations. The Neural network system identifies the commands by comparing them to patterns of possible operator commands included in the training file [13].

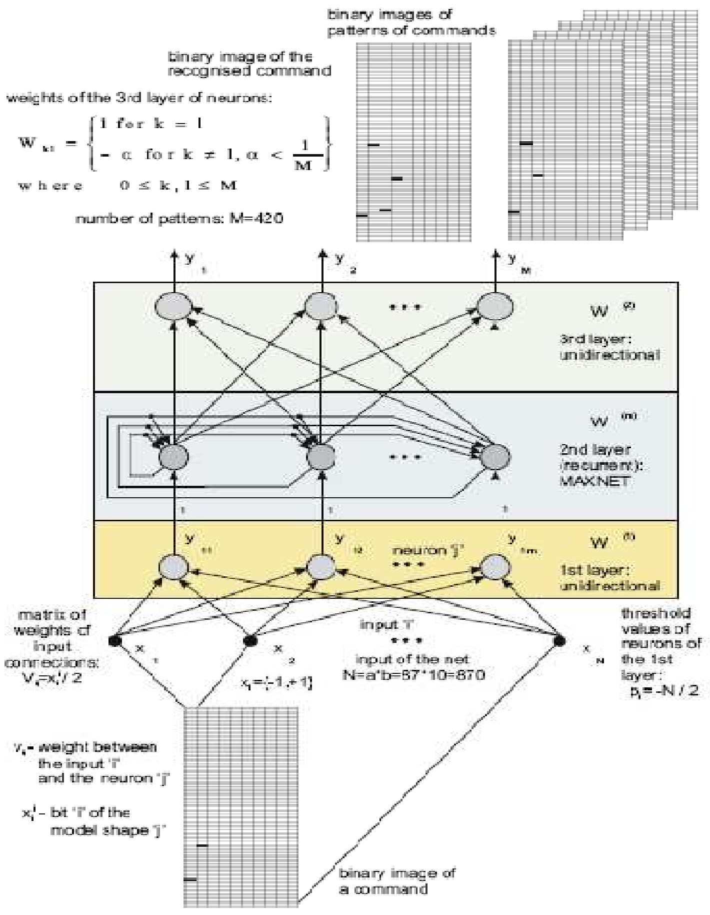

binary images of patterns of commands
binary image of the recognised command
weights of the 3rd layer of neurons:
$W_{kl} = \begin{cases} 1 \text{ for } k = l \\ -\alpha \text{ for } k \neq l, \alpha < \frac{1}{M} \end{cases}$
where $0 \le k, l \le M$
number of patterns: M=420
y_1
y_2
y_M
$W^{(2)}$
3rd layer: unidirectional
$W^{(m)}$
2nd layer (recurrent): MAXNET
y_{11}
y_{12}
y_{1m}
neuron 'j'
$W^{(1)}$
1st layer: unidirectional
matrix of weights of input connections: $V_{ij}=x_i^j/2$
x_1
x_2
x_N
$x_i=\{-1,+1\}$
input 'i'
input of the net N=a*b=87*10=870
threshold values of neurons of the 1st layer: $p_j = -N/2$
v_{ij} - weight between the input 'i' and the neuron 'j'
x_i^j - bit 'i' of the model shape 'j'
binary image of a command

Figure 2.5: 3-layer neural network for automatic command recognition [13]

This speech recognition method is used to correctly recognize 85-90% of operator's words. Tests have shown that more training of the neural network increases accuracy to around 95 %. However, this system lacks effectiveness when adding background noise. The recognition rate decreases from 86% at 70 dB noise power to 71% at 80 dB.

In order to enhance recognition accuracy, other neural network was used, based on sequence learning, to recognize speech [14]. The topology of the used network employs a number of sensors which are used to feed the input to the network and are entirely connected to the primary neurons. The network utilizes as much sensors as coefficients in the feature vector. Then, the primary neurons are connected to the second neurons. The number of second neurons is equal to the number of primary neurons which is itself equal to the number of feature vectors of a speech waveform. In addition, an adder is used to join the primary neurons and the previous secondary neurons. Afterwards, the result of the adder is stored in the present secondary neuron. Finally, the secondary neurons are connected to the output. The topology is demonstrated in Figure 2.6 [14]. For training, feature vectors corresponding to a speech waveform are used. The feature is extracted from the speech waveforms. Useful parameters, which are extracted from the speech waveform, are selected to represent the pattern efficiently. For this network, (Hidden Markov Model Toolkit) HTK [1] Fourier Transform based filter bank was used to give around equal resolution on a mel-scale. Training is used to update the weight values relating input sensors and the primary neurons for each word. Weights are calculated using Hebbian learning algorithm. During the recognition phase, the sensors take feature vectors as input. Then, sensors inputs and the connection weights to the neurons are used to compute values of primary neurons. If these values are negative, the primary neurons are set to zero. The new value stored in the secondary neurons corresponds to the values of the primary neurons added to the values of the pervious secondary neurons as shown below [14].

$$Primary = \begin{Bmatrix} 0 & if \quad I \times w1 < 0 \\ I \times w1 & if \quad I \times w1 \geq 0 \end{Bmatrix} \tag{2.3}$$

$$Secondary(t) = Primary(t) + \; Secondary(t-1) \tag{2.4}$$

The winning word for the input speech waveforms is chosen using the Winner Take All (WTA) [15].The network was tested for multi-speakers using one shot of training for a female speaker. Results have shown 100% recognition accuracy. The network was successful against distortion made by adding or removing part from the speech signal. Even with blurring signals, the results proved tremendous recognition. Moreover, the expertise proposed does not need long time for training.

[1] http://htk.eng.cam.ac.uk/ last accessed: 23 March 2010

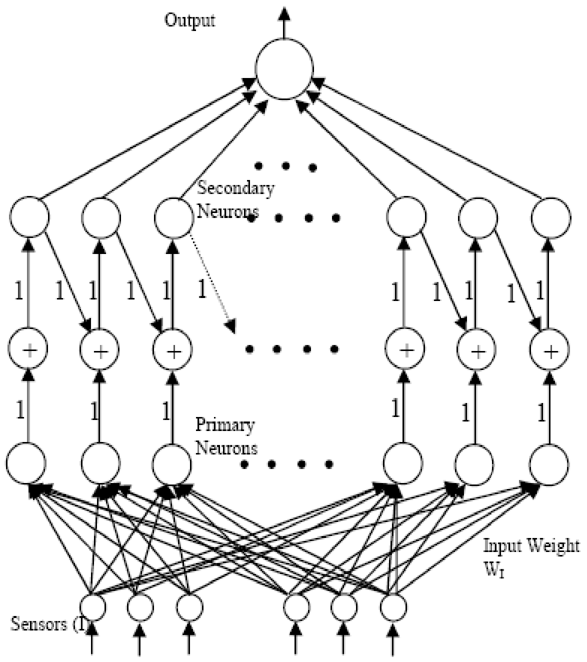


Figure 2.6 - Network Topology [14]

Besides the use of neural networks in a wide variety of tasks as speech recognition, other neural network was used to recognize and classify vowel signals into respective categories. In [16], a probabilistic neural network (PNN) model was used to classify speech signal. This PNN network is a feed-forward neural network that solves pattern classification problems. The latter applies the Bayes strategy for pattern classification in its leaning prototype. To categorize the speech waves of vowels, {a, e, i, o, u} and recognize them, the PNN is used and applied on an experimental setup. Sounds on reading vowels were recorded from 200 individuals of males and females. The speech waves are then stored in “wave” file format using standard sound recorder on a computer. A filtering process is also used to de-noise and compress signals. For filtering, Daubechies wavelet [17] approach was utilized. Finally filtered signals generate small number of approximation coefficients of speech waves which are then used for training and testing the PNN model.

Speech recognition setup is pointed up in Figure 2.7. For classification, the PNN model is implemented using the neural network Toolbox in MATLAB. To test the performance of the PNN model, a smoothing parameter was used called SPREAD to find out the effect of that parameter on the approach since that the latter is the most influential parameter affecting the performance. The PNN model has shown good results when adding the affecting parameter SPREAD. Accuracy of the network ended up between 68% and 100% concerning individual vowel classification and the overall accuracy was between 70% and 98 %. The PNN algorithm has produced motivating results for vowel recognition. However, this model is limited to only vowel recognition. More analysis is needed to develop a good network structure for speech recognition.

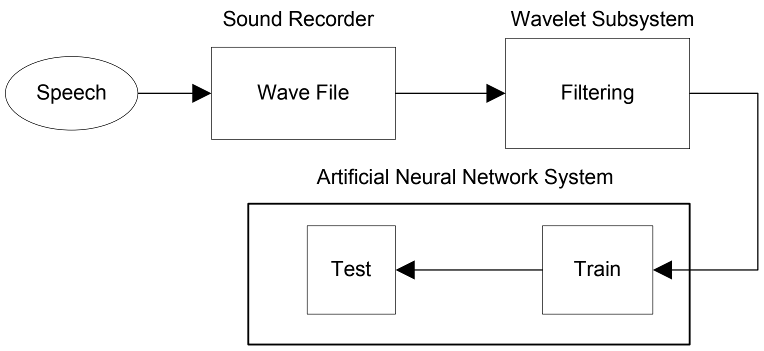


Figure 2.7 - Experimental setup for speech recognition [16]

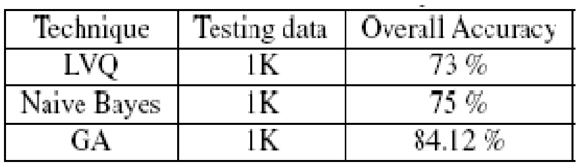
- **Other techniques**

In order to enhance the performance at the natural language understanding level, other techniques were used and some soft computing approaches were investigated. In [18], soft computing techniques such as Learning Vector Quantization and the Genetic Algorithm are developed to improve the performance of automated call routing applications and hence increase the accuracy of the natural language understanding of speech.

The proposed Genetic algorithm [19] approach consist of two phases. In Phase one, a naïve Bayes classifier is used to obtain probability tables for subjects and their intentions. The result is a 25*3 weight matrix consisting of a threshold, low-weight, and high-weight. Both filtered and non-filtered results were obtained.

However the approach produces better accuracy using filtered data. The second phase applies a (Genetic Algorithm) GA in order to increase the accuracy of the probabilities in the matrix computed in phase 1. This means that the system will be able to identify a given sentence and categorize it according to the training data. The GA attempts to obtain optimum values for threshold, low-weight, and high-weight and maximizes the following objective function: $argmax_j\, P(C_j\,|S)$ where $P(C_j\,|S) = P(C_j\,|w_1, w_2, ...w_n)$ and the P is the conditional probability of category Cj given a sentence input S. Additionally, the author tested the results obtained from the GA based approach against results obtained using (Learning Vector Quantization) LVQ algorithm. The LVQ was originally presented by Kohonen [20] showing some promising results. The algorithm represents labels as regions of data space paired with prototype vectors with the goal of assigning labels to high dimensional vectors.

Experiments results obtained (as shown in table 2.4) prove that the GA based approach outperforms LVQ and naïve Bayes. The results presented were obtained using 3k of data to train the system and only third of this as input test data (1k). However, the results do not present a fair comparison between LVQ and GA because the training data set is unbalanced. As a result, LVQ did not have reference data to identify a vector. Nevertheless, LVQ still produce satisfactory results. We also conclude that GA produces excellent results in case we do not have training data set. Another LVQ shortcoming using is the necessity for a large amount of training data.



Technique	Testing data	Overall Accuracy
LVQ	1K	73 %
Naive Bayes	1K	75 %
GA	1K	84.12 %

Table 2.4 Overall accuracy [18]

2.1.2 Recommendations

So far, we described and analyzed the various soft computing and intelligent systems techniques used for speech recognition. In addition, we discussed the use of fuzzy logic in speech recognition. Then we examined the use of neural networks in speech recognition and finally we explained some other techniques as LVQ and GA and their contribution in recognizing speech. These techniques are summarized in Table 5. We then present some recommendations to improve speech recognition quality and identify potential future works in the area. Table 5 summarized the techniques presented thus far in speech recognition. Although some techniques achieve encouraging results, this is usually at the expense of other factors, such as a longer training time. Moreover, some techniques tried to accomplish high accuracy rates in the presence of noise, with varying success. As a result of these observations, we believe an interesting approach to tackle the speech recognition problem would be one that combines both fuzzy logic and neural networks.

Science has improved drastically in fuzzy logic and neural networks over the past decade, and we can come up with a system that uses the advantages of both. We believe that a combination of these two would lead to a system that could overcome the dependence on expert knowledge and have the ability to mimic logical thinking as a result of its fuzzy logic aspect. At the same time, it could inherit the ability to self adapt and learn from neural networks.

Applying this to a speech recognition system, we could design a neural network that has multiple fuzzy layers for input variables, inference rules, etc. This could potentially lead to better accuracy rates as the system is able to self adapt and self learn to continuously improve its operation. However, this might require a longer training time due to the possibility of running into a local minimum problem.

Another possible technique to improve existing speech recognition systems would be through the use of genetic algorithms. We could use a GA, in a neural network, to find the best possible match to a given input speech signal. GA would lead to faster training and increased accuracy since the selection of inputs to the neural network from the large input set will be reduced to a minimum. Furthermore, a GA can be used to enhance the learning capability of a neural network, thus decreasing the speech recognition time and increasing the accuracy.

Technique	Accuracy	Details
Austermann et al. [4]	Speaker dependent mode - 84% Speaker independent mode - 64%	Recognition rate is similar to humans in speaker independent mode
Sun, Karray et al. [6]	81.76%	Recognition results tested 160 sentences
Alemzadeh et al. [9]	First Answer - 98% Third Answer - 99%	Average First Answer with noise - 90% Average Third Answer with noise - 96%
Halavati et al. [9]	First Answer - 85% Third Answer - 95%	Surpasses the HMM approach
Majewski and Kacalak [12]	85%-90%	More training increases accuracy to 95% Accuracy rate drops to 71% with 80dB of noise
Elmisery and Starzyk [14]	100%	Tested for multi-speakers using one shot of training Successful against distortion
Lim et al. [16]	70%-98%	68%-100% concerning individual vowel classification
Ullah, Karray et al. [18]	LVQ - 73% Naive Bayes - 75% GA - 84.12%	LVQ requires a large amount of training data GA produces excellent results with a small training data set

Table 2.5 Summary of presented techniques

2.1.3 Summary

In this chapter, we have presented a literature survey of existing soft computing techniques used in speech recognition systems today. We have focused on techniques that use fuzzy logic and neural networks. A few techniques presented also utilized genetic algorithms. As we can observe from the results, there is a wide variation in terms of the quality of the proposed techniques. Some achieve high accuracy values however at the expense of a longer training time. Some react differently to the existence of noise. Speech recognition is still a very challenging problem due to the difficulties associate with natural language understanding. It remains an open research problem and researchers are coming up with new techniques everyday due to its impact on different fields and future technologies.

Chapter 3

Human-Robot Verbal Interaction

3.1 Human-Robot Interaction

Human-Computer Interaction (HCI) refers mostly to Human-machine interaction. It represents the methods by which a user could interact with computers. Human-computer interaction is the discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them [21]. Robot technology and artificial intelligence are realizing increasing advances which have made robots applications used in versatile applications. The important growth in robots technology has also supported human interaction with robots. Human-Robot Interaction (HRI) is a subset of the field of Human-Computer Interaction. Human-Robot Interaction (HRI) represents how humans and robots could influence each other [22]. HRI is used in a variety of applications and tasks. It is employed in dangerous tasks such as urban search and rescue [23] . Robots are also used to assist elderly people [24] and the handicapped [25]. Human-Robot interaction requires dialogue management in order to realize the communication between both parties. Some metrics are also useful for evaluating the quality of the human-robot interface.

3.1.1 Dialogue management

As mentioned previously, interaction for human-robot systems is not a simple interaction. However, it requires dialogue management. Dialogue management refers to the communication between two or more parties. Both information and control are shared during the communication process. Information could be data, symbols and context. The structure or method of dialogue varies depending on the situation (task, environment, etc) [26]. Dialogue is usually arbitrated by an interface in order for humans and robots to communicate. Computer command languages are a kind of interfaces affording great power and flexibility, however they suffer from high cost learning. On the other hand, interfaces such as menus are easier for novice people to use because they don’t require great user knowledge. A good interface supplies architecture that assists human-robot dialogue and information exchange. The dialogue management performs by converting user input into understandable language to the computer and parsing the computer or robot language into user understandable language [27]. Different tasks must be handled within the dialogue management. Some of these tasks cover disambiguation, error handling, and role switching [28]. Role switching means that at any step of the dialogue, one of the contributors may have control of the conversation. Both robot and the user may be permitted to take the initiative of the conversation or toggle roles as required.

3.1.2 Human-robot interaction evaluating metrics

In this section evaluation metrics used to measure the efficiency of robots interacting with humans are discussed. The metrics focused on interface usefulness and human intention influence. In order to enhance the effectiveness of the human-robot interaction, several metrics are used. Metrics that help the design of human-robot interaction are described. Metrics include task effectiveness (TE), neglect tolerance (NT), robot attention demand (RAD), free time (FT), fan out (FO), and interaction effort (IE) [29].

3.1.2.1 Task effectiveness

Task effectiveness is used to measure how well tasks are achieved by human-robot teams. There is a distinction between overall task effectiveness and current task effectiveness. Overall task effectiveness is calculated after the robot accomplishes the task. An example illustrating this measure is to calculate the amount of time needed for the robot to finish the task. However, current task effectiveness calculates the success of the robot immediately. The success could be assessed by the robot speed to reach the final destination. However, the speed could be misleading. Sometimes the robot goes rapidly to the target but could reach an impasse and loose time to back out from it. In this situation the robot is making negative progress [29].

3.1.2.2 Neglect tolerance

The neglect tolerance (NT) is a significant metric to measure the autonomy of a robot given some task. The NT calculates the robot's current task effectiveness decrease over time after being neglected by the user. A neglect curve illustrates the effectiveness variation over the attention time for a given robot and a given problem space.

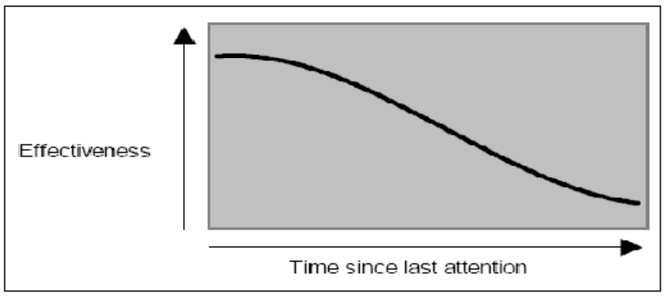


Figure 3.1 - Neglect curve [22]

The current task effectiveness decreases when the time since last attention of the user increases. Two ways can be used to measure the neglect tolerance. The first neglect tolerance measurement is called premeasured average neglect time. The latter can be computed by locating a robot- in a problem word containing obstacles and assigning to it an arbitrary task. The amount of effective time needed by the robot represents the first neglect tolerance measurement. This time depicts the elapsed time during which the robot gain distance before falling below the effectiveness threshold. The second method is based on the human's estimate of task progress. The neglect tolerance is measured by the actual active usage of the robot by the user. It represents the time between some user instruction and other new user instruction. In order to increase the neglect tolerance of a robot, one solution could be increasing its intelligence and autonomy. Neglect tolerance is one of the steps to enhance the human-robot team. However, other metrics are needed to ensure good human-robot design [22].

3.1.2.3 Robot Attention Demand

Robot attention demand (RAD) computes how much attention is needed by the robot. It represents the total user attendance time given to the robot. RAD is represented as a correlation between NT and the interaction effort (IE). The interaction effort represents the total time needed for the user to interact with the robot.

The association of these three variables is defined as follows:

$$RAD = IE/(IE + NT) \tag{3.1}$$

RAD is a fraction of the time absorbed when the human interacts with the robot. The numerator is the amount of the time that the user must spend behaving with the robot. The denominator is the total amount of effective time of the robot. In order for the user to handle and focus on other things while behaving with the robot, the RAD must decrease. Reducing the RAD is one of the keys to build good human-robot interface. RAD decreases when either increasing NT or decreasing IE. Increasing the NT is not always an efficient solution to diminish RAD because of the dependency relation that combines NT and IE.

Increasing the NT is not always efficient for restraining the RAD. Creating autonomous robots is one example justifying that. Autonomous robots are not easy to conceive since they require advanced engineering and programming. These kinds of robots could be practically autonomous with a high NT. However, improving the robot performance entails its re-engineering and reprogramming which is a sort of interaction that costs great amount of effort. Hence, the time expended by the user to interact with the robot is greater than the time needed for the robot to perform autonomously [22].

3.1.2.4 Free Time

The free time is a metric related to the RAD to calculate the amount of time that the user does not need to pay attention to the robot. The free time is useful to measure the RAD. The user could use the free time to perform other surrogate task. In a human-robot environment, the user may pay attention to other tasks in addition to the robot. These tasks may include surveillance, finding victims of a disaster or investigating the field. Measuring free time is not an easy task because we do not know when the user is free. The free time can be detected by reducing the RAD. The measurement of the free time is not important. In fact, changes in human-robot interface are more important since they provide information about the increase of FT and the decrease of RAD.

3.1.2.5 Fan-out

Human attention can be managed by allowing the user to control several robots simultaneously. This allows the human to perform additional tasks quickly and successfully. Many robots can protect a space more effectively than single robot in tasks such as surveillance or search and rescue. Effectiveness of a human-robot team can be measured using the fan-out. Fan-out is used to assess the number of robots that a user controls at once and efficiently. The fan-out metric [22] is defined as:

$$FO = 1.0/(RAD\quad) = (IE + NT)/IE \tag{3.2}$$

Referring to this equation, the FO increases as neglect tolerance becomes larger comparing to interaction time. As the neglect tolerance increases, the user can operate more robots. The fan-out is limited by human cognition and primarily memory constraints. The human should memorize robot state information, interface modes, robot abilities in order to handle multiple robots. Remembering these information demands advanced memory work since the human can store only restricted amount of information in the short-memory and few mental models can be active in long-memory at a time. These restrictions could affect the fan-out.

The most common goal for building human-robots interfaces is to increase the effectiveness of the team in achieving some task. Studies in human-robot metrics have shown that increasing the neglect tolerance of the robots and decreasing the interaction effort of the interface are clues to establish effectiveness. This was explained in the free-time and fan-out metrics. These metrics coupled with neglect tolerance can be computed and therefore used to generate estimates of interaction effort. Hence, progress in improving human-robots interfaces can be assessed.

3.2 Human-Robot verbal interaction

Verbal communication is considered a Natural User Interface (NUI) [30], which relies on speech synthesis and recognition. Such type of interfaces are very important for a broad range of users such as hands/eyes-busy users, motor-impaired users, users with vision impairment and users working in hostile environments. Verbal interaction is very popular in robotics especially in personal assistive robots, which are used to help elderly people and in entertainment robots. Several research endeavors have been assigned to endow the robots with verbal interaction as a high-level faculty [31] [32] [33]. Even though the language usages of many of them were simple and may not be considered as full speech dialogue systems. For example, RHINO

[34] is a guide robot used in a museum to characterize particular shows. Though the robot cannot carry true speech dialogues, it is able to recognize commands like “execute tour number 3”. Using speech utterances, the robot handles tours within the museum and follows predefined routes. Similar to RHINO, Polly [35] was used in office environments to offer guided tours. Interaction comprises vision detection in addition to the speech input. In order for the user to go through a tour, he waves his feet. Then, the robot will guide him through a tour using built speech utterances and describes diverse landmarks.

MAIA [36] is another robot that could take objects from one place to another. The robot also responds to simple user spoken commands expressions. Related to this research work, a mobile office assistant was created at the Microsoft Research Institute of Macquarie University. The robot can distribute packages, direct visitors to offices, and supplies guided tours. To recognize speech, the robot implements a state-based dialogue and a language analyzer [37]. Jijo-2 [38] [39] has similar abilities to the mobile office assistant robot. It can communicate information and lead people through an office environment. It differs from other office assistant robots by using Japanese speech dialogue system to communicate.

AESOP 3000 surgical robot [40] is another specific example of assistant robots. This robot is adapted to sensitive heart surgery applications. The robot plays the role of the hand of the surgeon. The latter commands the robot using voice commands. This robot presents an encouraging example among the rising use of robots based on natural language interactions. However the project did not carry a full speech dialogue system. Other robots are also created enclosing multi-modal interfaces that include speech, keyboard and point-and-click input.

Generally speaking, human-robot verbal interaction varies from understanding simple commands to extracting all the information in the speech signal such as words, meanings and emotional condition of the user. To develop an interface with natural language understanding ability, many issues must be taken into account such as dealing with the ungrammatical nature of many spoken utterances, the detection of errors in speech recognition and interpretation, and the design of intelligent clarification dialogues. The realization of the potential of verbal interaction also requires the solution of other challenging problems such as dealing with frequently and varying background noise, word echo and repetition, and also with different and background sound sources overlapping the speech [3]**.** For example, to figure out exactly what a user wants to say, many automatic speech recognizers (ASR) have been developed with varying capabilities. An automatic speech recognizer system, such as Microsoft SAPI 5.1 [41]**,** provides 95 to 97 percent accuracy in dictation experiments, where user training is obtainable and misrecognized words out coming errors are intercepted. However, for an assistive robot even a small percentage of misrecognized words could generate troubles and present a high risk for different applications. This is illustrated in the robotic guide for blind [42].

Using Automatic Speech Recognition (ASR) systems to have visually impaired users communicate with robots was not an effective solution for speech recognition. In fact, many speech recognition errors happen when the person directed by the robot stops and starts conversation with someone else [43]. The speech recognition parses words continuously and may recognize wrongly phrases said by the guided person to other interlocutor as route directives. Hence, the robot moves in wrong direction. The ASR system could also interpret garbage words such as throat clearing or coughs sounds as directives to move to other places. Other problems that could affect the ASR, employed in human-robot natural language dialogue, are pronounced speech defects unrecognized by the ASR. The origin of these speech defects are particular physical disabilities such that spinal cord injuries.

3.3 Summary

Training, high level perception and planning capabilities could be a solution for robots to be able to understand a variety of verbal instructions. However, training may not be practical, because it is extremely hard to find a representative sample of the target users. For instance, it is hard to gather typical sample from the population of the visually impaired users of a robotic guide used at an airport [43]. More clearly, the obligation for the target user to admit training may be a serious concern.

Chapter 4

Proposed Approach

This chapter presents the different components of the proposed system for natural language understanding in details. We start by presenting the proposed system architecture. Julius, the used Automatic Speech Recognizer (ASR) is then explained followed by discussing the graphical user interface of the system.

4.1 System Architecture

4.1.1 System Description

The proposed system is mainly divided into two major components, namely, Julius Speech Recognizer engine module and Java application module as illustrated in Figure 4.1. The user interacts with the robots via the Java-based Graphical User Interface (GUI). The user acts on robots using natural speech commands. The speech input is recorded using the java application. The user starts recording his speech utterances. When the user stops recording, the java application saves his speech input to be archived in a" .wave" file. The ".wave" file contains speech input from the user. This wave file is then parsed by the Julius automatic speech recognizer engine.

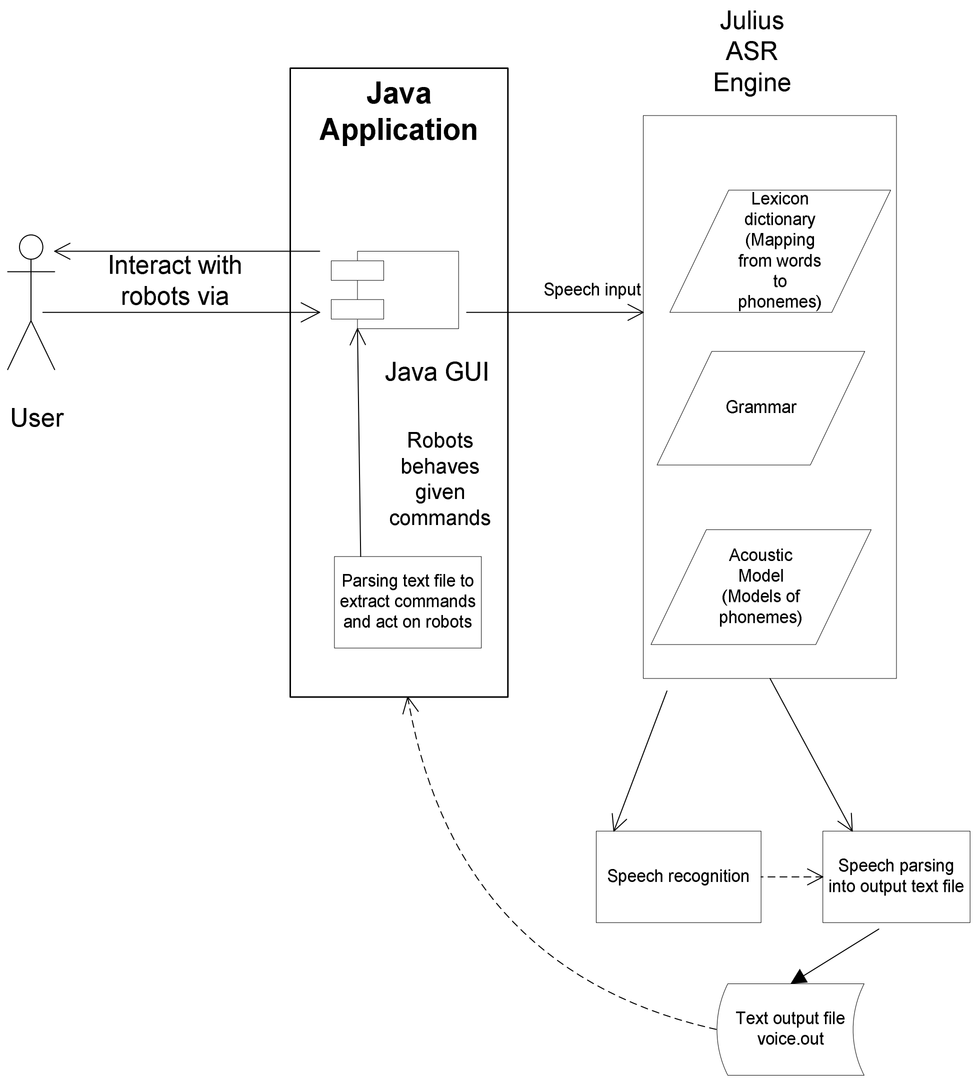

Julius ASR Engine
Java Application
Lexicon dictionary (Mapping from words to phonemes)
Interact with robots via
Speech input
User
Java GUI
Grammar
Robots behaves given commands
Acoustic Model (Models of phonemes)
Parsing text file to extract commands and act on robots
Speech recognition
Speech parsing into output text file
Text output file voice.out

Figure 4.1 - System architecture

The Julius system, as illustrated in Figure 4.2, is composed of grammar, lexicon and an acoustic model. We have altered the grammar module as explained previously to adapt the recognition to our natural speech system. The recognized wave file is parsed into an output text file containing the recognized words from user utterances. The output file ".out" is then used by the java application to extract commands. These commands act on the robots within the Java GUI. In order to present the human-robot verbal interaction, we have simulated robots' actions on a Java GUI. The simulation demonstrates clearly how the robots are behaving to the commands coming from user speech input. The robots behave to the commands extracted from the ".out" text files. The java application parses the ".out" text file to get user commands. The following subsections describe in details two major components of the system.

4.2 Julius ASR System

The Julius automatic speech recognizer engine was used in our system in order to parse the speech input of the user. Julius [44] is an open source software that recognizes speech continuously using word N-grams. Julius performs high recognition rate coupled with high-speed speech recognition. The recognition rate could exceed the 90% for a 20,000-word vocabulary dictation task. In addition to that, Julius performs at almost real time. This makes it appropriate for use in several human robot tasks. The Julius system is composed of a language model, acoustic model and a pronunciation dictionary. This combination affords one user the possibility to employ Julius in order to build versatile tasks for specific systems. Also, the Julius code is open source so that it was able to alter the code or recompile the engine for our specific natural language use. The Julius system overview is illustrated in figure 4.2. The acoustic model represents the models of phonemes. The language model contains a very large list of words and their probability of occurrence in a given sentence. The lexicon dictionary characterizes the mapping from words to phonemes. In our thesis, we have used the Julius system for natural speech recognition. We modified the grammar component of Julius for building our human-robot verbal interaction system. Julius achieves recognition on microphone input (live mode) or audio files (archive mode). In our thesis, we have used the audio file mode.

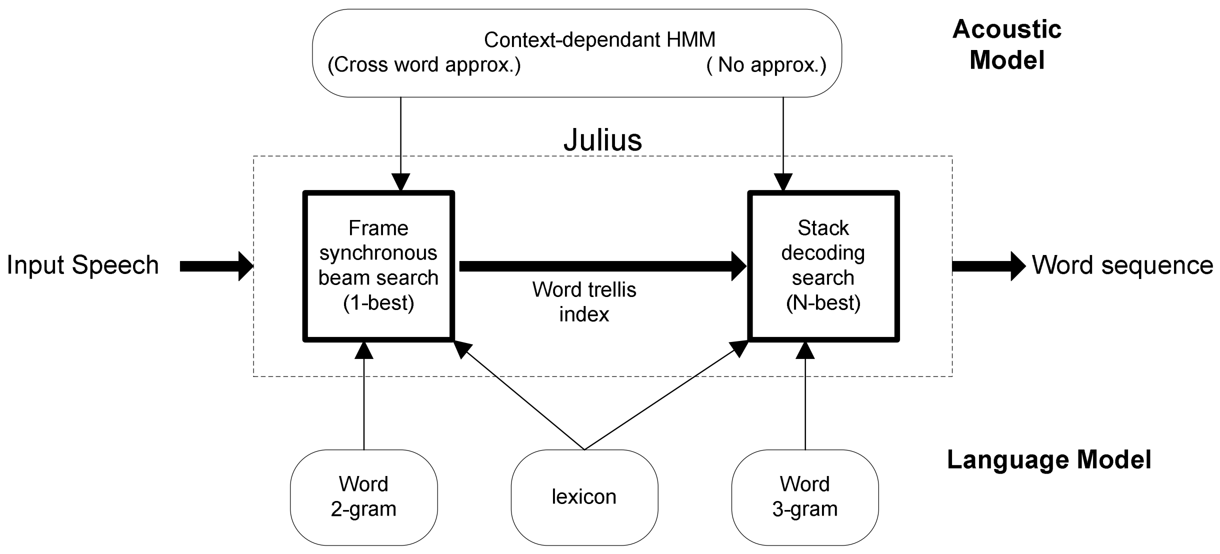


Figure 4.2 - Julius Architecture [44]

4.2.1 Grammar Module

We have altered the grammar module of Julius to adapt it to our natural speech understanding system. Grammar files are much smaller than language modules mentioned above. The grammar file [45] contains sets of predefined combinations of words. Each word in the grammar file is mapped to list of corresponding phonemes. Phonemes represent the distinct sounds that create up a word. The recognition grammar describes the list of phrases or words that the speech recognizer engine listens for and accepts. Once one of these words is heard, the engine returns these words to the calling program. In our thesis, we have generated a new grammar file containing the words that the system needs to achieve natural speech recognition to be used for further action in robots. For that we have used the predefined and built Acoustic model as well as the lexicon. We have used the words needed for our system. These words are already trained in the acoustic model and exist on the vocabulary of the Julius engine. The recognition grammar is composed of two files. The "**.**grammar" file contains a set of rules to present the words that the Speech recognizer engine is expecting to recognize. The “.grammar file” does not include all the words to be recognized. Instead, it contains word categorizes. These word categories represent the name for a list of words defined in the second file. The “.voca” file describes the list of words corresponding to each word category and defines the pronunciation information for every word. A modified Backus–Naur Form (BNF) format is used to define the rules leading the accepted words. The .grammar specification employs a set of derivation rules presented as:

```
Symbol: [expression with Symbols]
```

The symbol is non terminal which means that it can be represented in terms of other symbols. [expression with symbols] symbolizes an expression that encloses sequences of symbols. These symbols could be non terminal or terminal. A terminal symbol is a constant value which never appears to the left of the colon. Terminals in “.grammar” files represent word categories that are defined in the “.voca” files. This is the derivation rule we have used for the “.grammar” file:

```
S: Sil1 Sent Sil2
Sent: robot Num Mov Obj
```

“S” represents the initial sentence symbol. “Sil1” and “Sil2” represent the silence that happens before and after the user utterance to recognize. “Sil1”, “Sil2”, “robot”, “Num”, “Mov” and “Obj” corresponds to the word categories which will be defined in the “.voca” file later.

Word definitions corresponding to the word categories explained above are presented in the “.voca” file.

To define the word category, we precede it by the “%” symbol. Afterwards, word definitions are added one by line for every corresponding word categories. For each line, the first column is mapped to the string referring to the output that will be recognized. The second column represents the pronunciation of the word. Figure 4.3 illustrates the “.voca” file adapted to our natural interaction system:

```
% Sil1
<s> sil
% Sil2
</s> sil
% robot
robot r ow b ao t
% Num
one hh w ah n
one w ah n
two t uw
% Mov
move_to m uw v t uh
move_to m uw v t ah
move m uw v
move m uh v
pick_up p ih k ah p
go_back g ow b ae k
hand_me hh ae n d m iy
put_back p uh t b ae k
put_back p uw t b ae k
% Obj
cup k ah p
place p l ey s
```

Figure 4.3 - The ".voca" new file

“sil1” and “sil2” are described using the words <s> and </s>. Each word definition is labelled using the silence models “sil” within the acoustic model of the speech recognizer. The word category “Num” is, for instance, divided into “one” “two” “three” and their matching pronunciation information. This file could be modified dependably of what is expected from the system to recognize. We may add words and delete garbage useless words to be recognized. However, words must be included within the dictionary of the recognizer engine. Every added word definition must be preceded by its conforming pronunciation information which could be found in the “dict.all” file. The mkdfa.pl script grammar compiler is then used to compile both “.voca” and “.grammar” files. More clearly, the compilation is done by looking for the symbol “S” first in the grammar file and by then replacing each word category with all the word definitions in the “.voca” file. After compilation, we end up with two files “.dfa” and “.dict”. These files will be employed later for recognition of user voice commands.

4.3 MVC-based Graphical User Interface

As mentioned previously, the user interacts with the robots via Java-based Graphical User Interface (GUI). This interface has been designed based on the use of design patterns in order to provide flexible user interfaces with minimal coupling relationships between the components. Design patterns help software reuse, and allow the description of the structure and the collaboration scheme between components of high abstraction level. In order to design our application, many software design patterns could be used. The Model Viewer Controller (MVC) design pattern was used to architect our application. The MVC [46] is used to develop loosely-coupled applications. MVC had showed effective results for applications based on GUIs. The basic idea of the MVC model is to decompose the application into three parts as shown in Figure 4.4. The model contains all enterprise data and business logic needed to process data. The view handles the graphical output and depends on the model. It is updated whenever the model changes. The view visualizes the state of the model. Finally the controller is composed of objects that receives user actions and transfer them into requests to be processed within the model. The controller interprets user input that will be performed by the model.

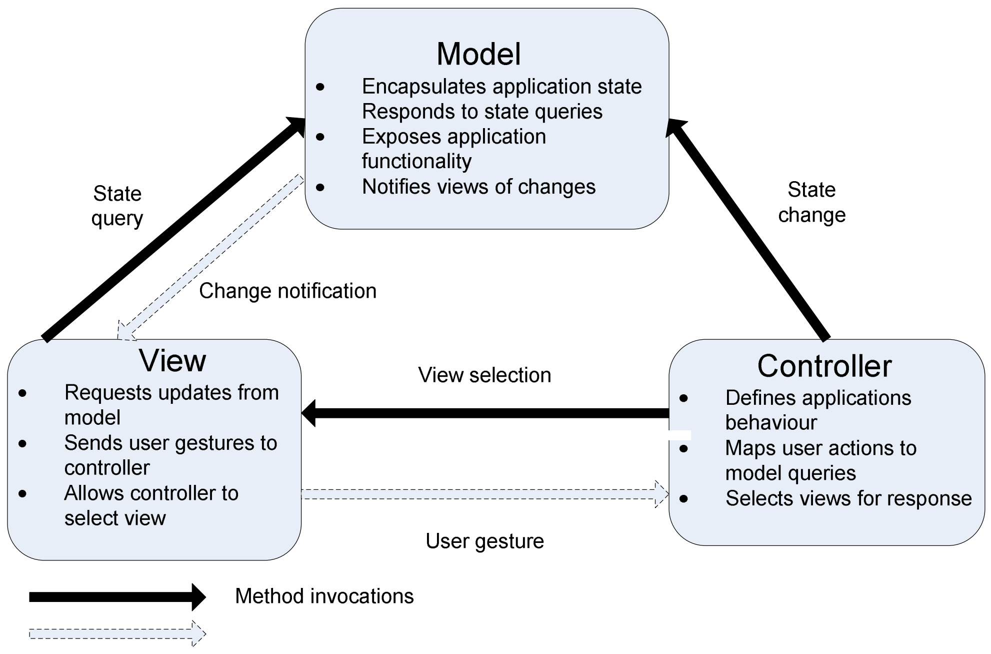

Model
Encapsulates application state
Responds to state queries
Exposes application functionality
Notifies views of changes
State query
State change
Change notification
View
Requests updates from model
Sends user gestures to controller
Allows controller to select view
View selection
Controller
Defines applications behaviour
Maps user actions to model queries
Selects views for response
User gesture
Method invocations

Figure 4.4 - MVC architecture [46]

Many advantages in the MVC architecture have made it appropriate for building our application. The system could include many views using the MVC architecture. There is no processing in the view, it represent data for the user. In addition, the data returned from the model could be displayed in many views. This allows the same code to be used in many other different applications. Hence, code duplication is avoided. The controller also allows the user to get input without processing it and requests to the corresponding model for execution. The utilized design pattern facilitates the reuse of our application in different tasks or fields. The MVC makes our application robust, allows the code to be reused for new applications. The system is well organized using such a design pattern.

An abstract representation of our MVC design pattern could be described as following:

The controller listens to events from GUI and starts or stops recording given the user gestures and input. Then, the controller sends recordings and natural voice input to be recognized by the Julius Server. Afterwards, the model parses the output text that contains recognized commands for robots and issues these commands to the controller. The latter updates the GUI by simulating the movement of robots on the interface. In addition, the user gestures on buttons are sent from the View to the controller to define the application behavior. A more detailed demonstration of the model is depicted in the following figures.

Figure 4.5 illustrates the view's components of our application.

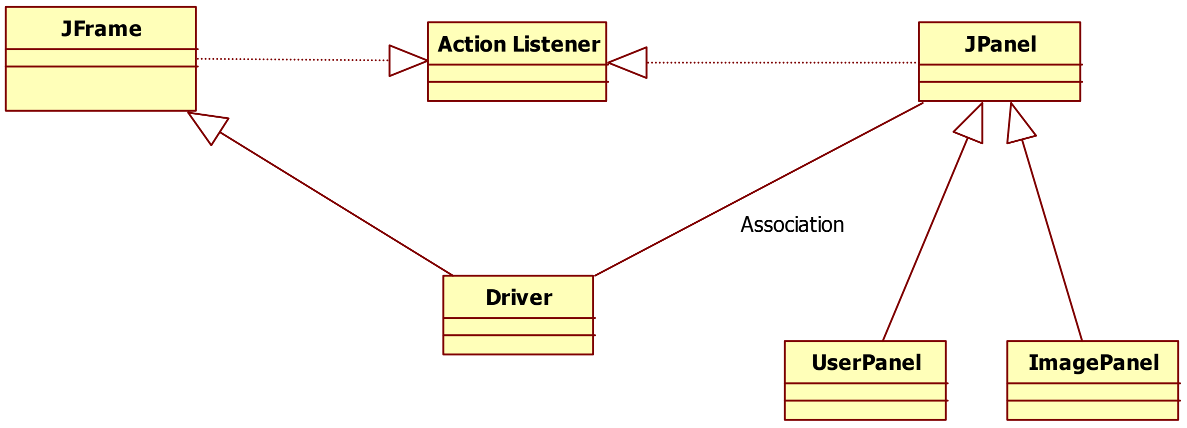


Figure 4.5 - The view

The controller is showed in the following Figure 4.6.

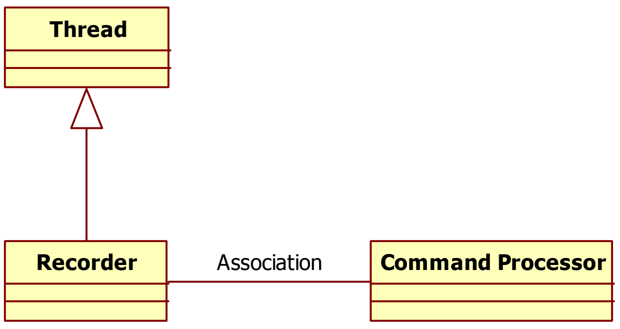


Figure 4.6 - The controller

The model, as explained previously, encloses the processed data and business rules. In our system, the model also contains wave files and ".out" parsed text files. The different classes showed above will be explained in the UML class diagram.

4.3.1 Use Case Diagram

Use case model [47] describes ways in which an application is to be used. Requirements are often naturally expressed as an interaction between application and user. A use case is represented by an interaction between actors and the application. The actor consists of a user or an external application in some cases. Figure 4.7 shows a use case diagram illustrating our system behaviour and how the user in interacting with the application.

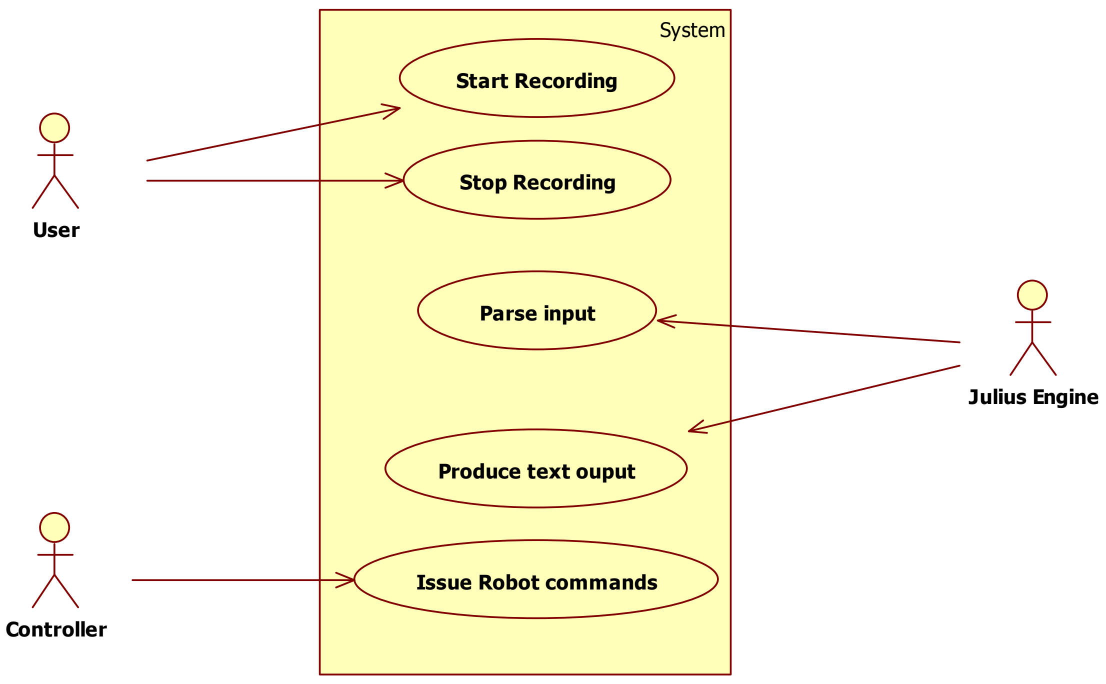


Figure 4.7 - Use Case Diagram

4.3.2 Class Diagram

The class diagram shows class elements and how they are connected via relationships within our application. Used class attributes and methods are described in the following Figure 4.8.

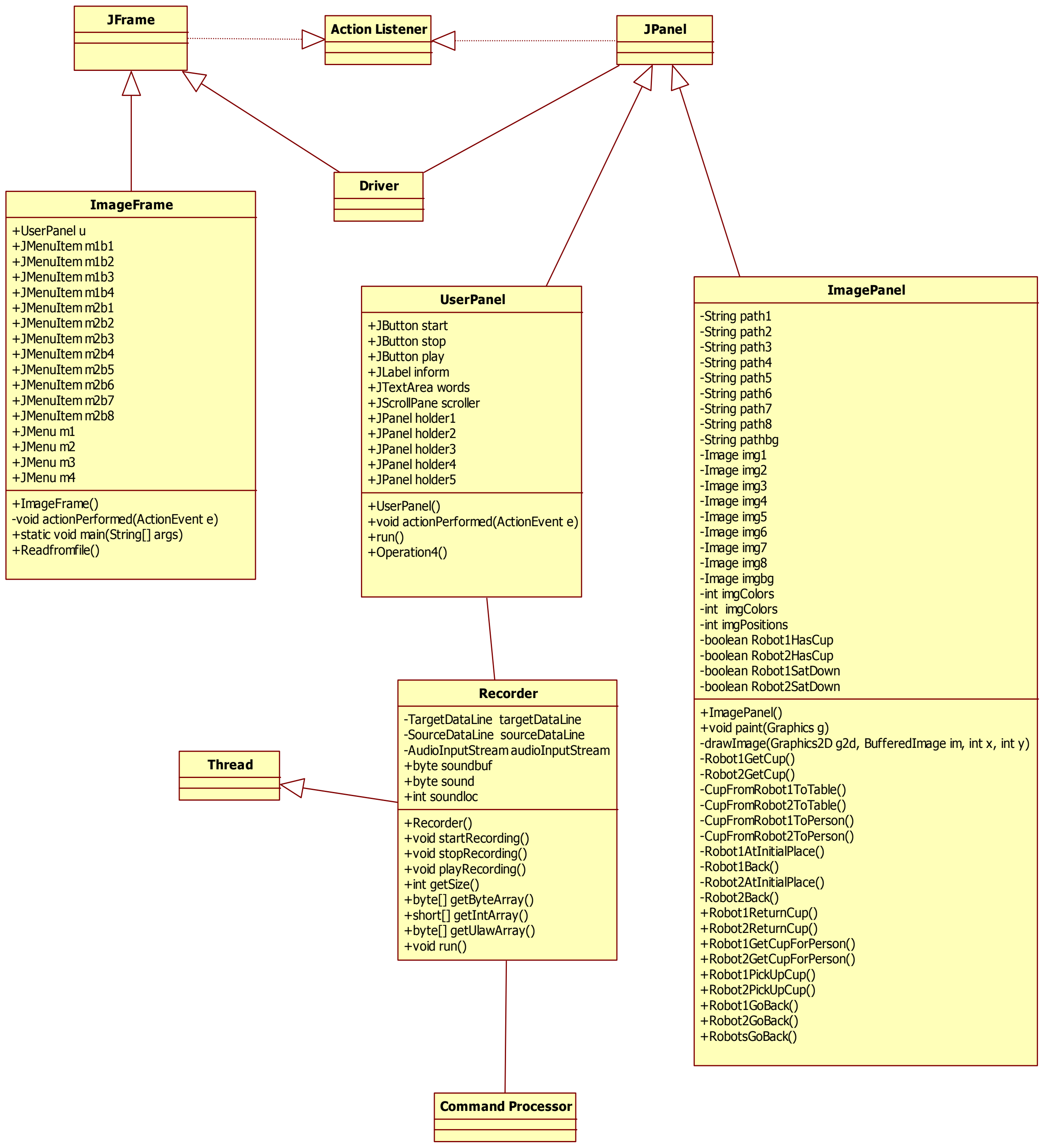


Figure 4.8 - Class Diagram

The most important methods used within the different application’s classes are detailed below:

- *ImageFrame class*

Methods	**Descriptions**
ImageFrame()	This method represents the constructor of the class. Within this method, a new user panel object is declared
Void actionPerformed(ActionEvent e)	This method is used to represent the actions performed on the menu item
Readfromfile()	In this method the text output file is parsed and commands are issued to act on robots
static void main(String[] args)	This method is the main used to run our application

- *ImagePanel Class*

Methods	**Descriptions**
ImagePanel()	This method represents the constructor of the class.
void paint(Graphics g)	This method is used to draw the graphic of the image panel and draw the images within the panel as well
Robot1GetCup()	Using this method, robot1 gets the cup from any other place such as the table or from a person
Robot2GetCup()	Using this method, robot2 gets the cup from any other place such as the table or from a person
CupFromRobot1ToTable()	In this method , robot 1 puts the cup back on the table
CupFromRobot2ToTable()	In this method , robot 2 puts the cup back on the table
CupFromRobot1ToPerson()	In this method, robot1 hands the cup to the person. This method calls other methods in order to hand the cup to the person. Many steps are executed to hand the cup to the person. First, robot 1 goes to the person and then he give him the cup
CupFromRobot2ToPerson()	Same as pervious method but robot1 is replaced by robot2
Robot1AtInitialPlace()	This method is used to check if robot 1 is in its initial place or not
Robot2AtInitialPlace()	This method is used to check whether robot2 is in its initial place and return true or false
Robot1Back()	This method is used to return robot1 back to its initial place
Robot2Back()	This method is used to return robot2 back to its

	initial place
Robot1ReturnCup()	This method tells robot 1 to put the cup back onto the table
Robot2ReturnCup()	This method tells robot 2 to put the cup back onto the table
Robot1GetCupForPerson()	Using this method, robot 1 gets the cup form the table and gives to the person. In this method, there is a call to other methods such as *Robot1getcup()*that allows the robot to get the cup from the table and *CupFromRobot1ToPerson()* that tells robot1 to give that cup to the person
Robot2GetCupForPerson()	This method is similar to the previous one but robot 2 is acting instead of robot1
Robot1PickUpCup()	Using this method, robot1 picks up the cup. In this method there is a call to the method *Robot1GetCup ()* and a SimpleAudioPlayer object is instantiated to play the audio message
Robot2PickUpCup()	The same behaviour but robot2 is used instead of robot1
Robot1GoBack()	It tells Robot 1 to go back to where it initially was. This method calls *Robot1Back()*
Robot2GoBack()	It tells Robot 2 to go back to where it initially was. This method calls *Robot2Back()*
RobotsGoBack()	This method tells Robot 1 and Robot 2 to go back to where they initially were

- *UserPanel Class*

Methods	**Descriptions**
UserPanel()	This method represents the constructor of the class. A new Recorder object and a new ImagePanel object are instantiated within this method
void actionPerformed(ActionEvent e)	This method is used to execute actions performed in the panel menu

- *Recorder Class*

Methods	**Descriptions**
Recorder()	This method represents the constructor of the class. An AudioFormat object is created and a DataLine object as well. These objects are used for recording input voice and set up the format.
void startRecording()	This method is used to start recording voice

	from micro input
void stopRecording()	This method is used to stop the recording. In this method there is a call to Readfromfile method. After stopping the recording, the User speech input is recognized and parsed to text commands that will act on robots in the GUI simulation

4.3.3 Sequence Diagram

The sequence Diagram was used to show the communication between the different instances of the application. The communications are partially ordered in time as shown in Figure 4.9. This Diagram shows interactions between different instances in the model. These interactions are represented using different links and stimulus.

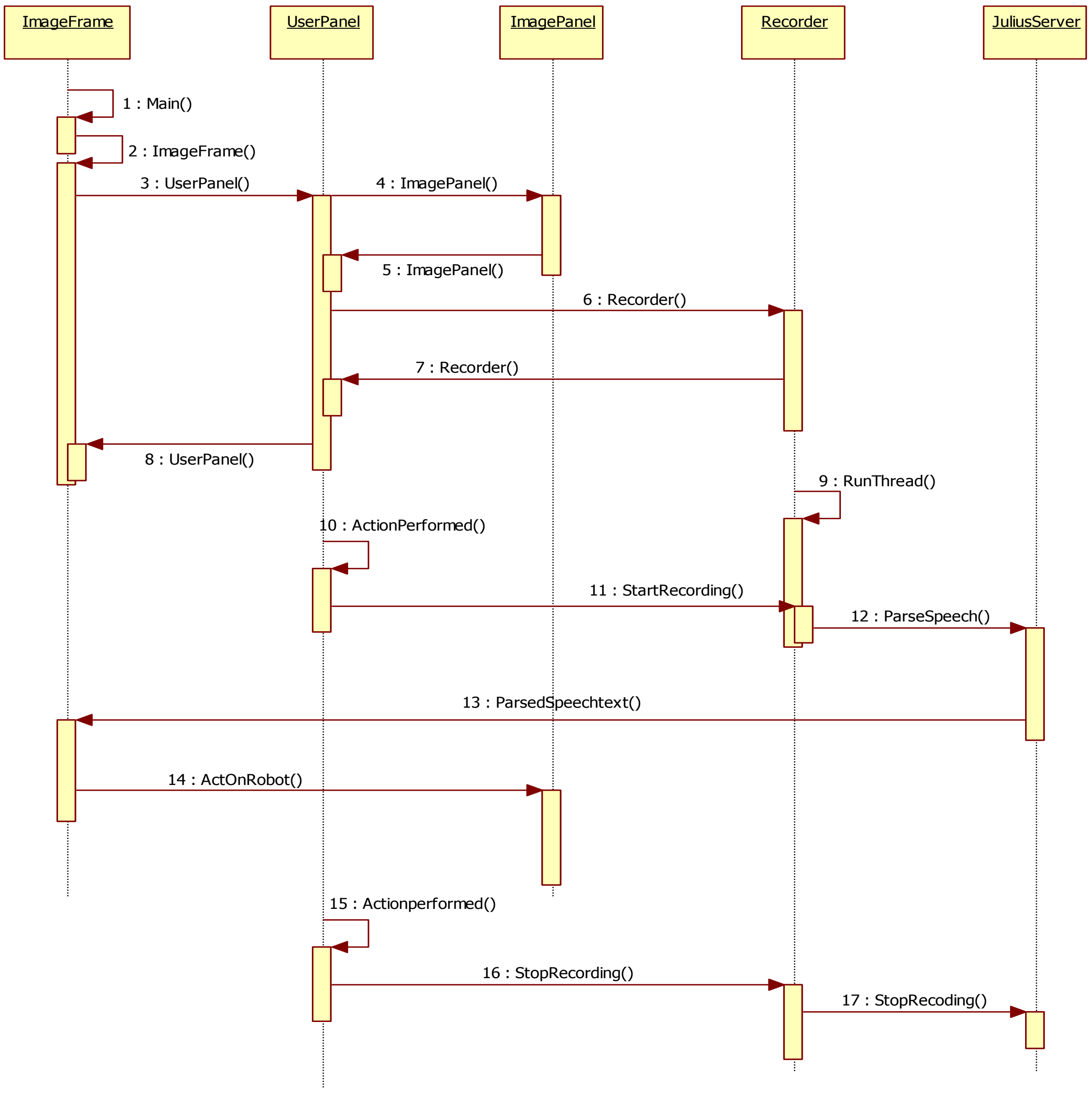


Figure 4.9 - Sequence Diagram

4.4 Summary

In this chapter, we presented our proposed system architecture. The architecture consists of three main components, namely the Julius ASR, the grammar module and the graphical user interface. We also explained in details how these components interact with each other to achieve our goals.

Chapter 5

Experimental Results and Discussion

This chapter presents the main experimental results that have been used to quantitatively evaluate the performance of the proposed approach. In this Chapter, experiment setup is described followed by highlighting the evaluation measures used. Quantitative results are reported and discussed in this chapter.

5.1 Experimental setup

In order to test and find out the robustness of our system, several experiments have been conducted and analyzed outcome results. The system is trained by using pre saved audio wave files. This system gets the audio files as input and produces results given different variables variations within these files. We have created 67 different audio file. Each file contains a natural voice wave command from a human natural voice. Diverse range of audio files was used to carry out specific experiments which rely on fixing a particular variable to calculate the appropriate measure explained in the following section. We have varied the voice speed of some of the commands. We have also included background noise as a variable to test if the system is recognizing commands and the robots are acting correctly even in a noisy environment. One of the variables also was the number of speakers. We have then saved some audio files using more than 1 speaker.

We have also altered the lateral distance from the microphone to show how the system is robust when changing distance.

5.2 Evaluation metrics

The following evaluation metrics have been used to quantitatively assess the performance of the proposed system:

- **Word Correct Rate**

The Word Correct Rate (WCR) was used as a performance measure for speech recognition. This measure was tested against several variables to investigate the robustness of our natural language understanding system. The Word Error Rate (WER) [48] could be used for this purpose. However the meaning of the absolute value of this WER is difficult to interpret or to compare to different rates. WER is defined as follows:

$$WER = \frac{S+D+I}{Nr} \tag{5.1}$$

Where

Nr is the total words in our reference audio file, S represents the number of substituted words after recognizing the audio file, D is the number of words from the reference audio file deleted in the parsing session and I as the number of words added to the parsed file and not appearing in the reference audio file.

One of the major problems of the WER was the normalization by Nr. In fact, the numerator of the WER presents a drawback as it contains insertions. The numerator is not bounded by [0, Nr]. Hence, the WER might surpass unity or might be negative. These disadvantages have made the WER an imprecise evaluation metric due to the difficult interpretation of its absolute value. Instead, we have employed the Word Correct

Rate (WCR). Contrary to the WER, the WCR has shown promising results and precise absolute rate value. The numerator is limited by the value of its denominator.

WCR is defined as follows:

$$WCR = \frac{H}{Nr} \tag{5.2}$$

Where H corresponds to the number of correctly recognized words [48].

- **Recognition Time**

Recognition time is also used as a measure to evaluate the system. The recognition time in our system consist of calculating the time needed for the Automatic speech recognizer (ASR) to recognize the audio file and parse it into an output text file which will be then used to act on robots. This ASR can be working either on a live or archive mode (file mode). In our method we have used the file mode to simulate the natural language understanding. The recognition time was computed based on the archive mode.

- **Success and False Action Rate**

In order for us to evaluate how the system is accurate when using natural language commands, we have used the success and false action rate. By Success Action Rate, we meant that the robots acts correctly as expected in response to the audio file command. On the other hand, the False Action Rate corresponds to the action of the robot to the audio file command, however in a wrong way. It is a false mapping of the command. More precisely, the robots acts to the command but not as expected. Here are some examples illustrating both the Success and False Action Rate:

Audio file command: **Robot one pick-up cup**

Simulation response in the GUI: The first robot pick up the cup as expected

➔ *The Success Action Rate for this example is 1*

Audio file command: **Robot one pick-up cup**

Simulation response in the GUI: The first robot acts but not correctly. It might for instance hand the cup to the person instead of picking up the cup.

➔ *The False Action Rate for this example is 1*

If there is no action to the audio file command both *True and False Action Rate are equal to 0*

In the next section we will discuss how these evaluation measures were computed against the variables.

5.3 Results and Discussion

5.3.1 Word Correct Rate

The audio command files created previously are now used to evaluate the system output. We calculate the WCR against one of the variables explained above. To calculate the WCR, we feed 9 audio files to the system without background noise and 9 other audio files with background noise. Each audio file of the 9 contains the same command. Hence, we run the same command (*Robot one pick-up cup*) 9 times without background noise and we run it 9 more times with background noise.

Table 5.1 shows the result of the WCR without and with adding the background noise to the audio commands.

Command1	WCR without background noise	WCR with background noise
1	1	0.75
2	1	1
3	1	0.75
4	1	1
5	1	0.75
6	1	1
7	1	0.75
8	1	0.75
9	1	0.75

Table 5.1 Word Correct Rate with and without background noise

For the audio file command created without background noise, the system has showed good results. The word correct rate was 1 for all the audio file instances as shown in Table 5.1. This rate decreases slightly for some audio files with background noise. The system recognized 3 out of 4 words. It is clear that the system recognized the majority of words even when adding background noise to the audio command. Figure 5.1 illustrates these output results.

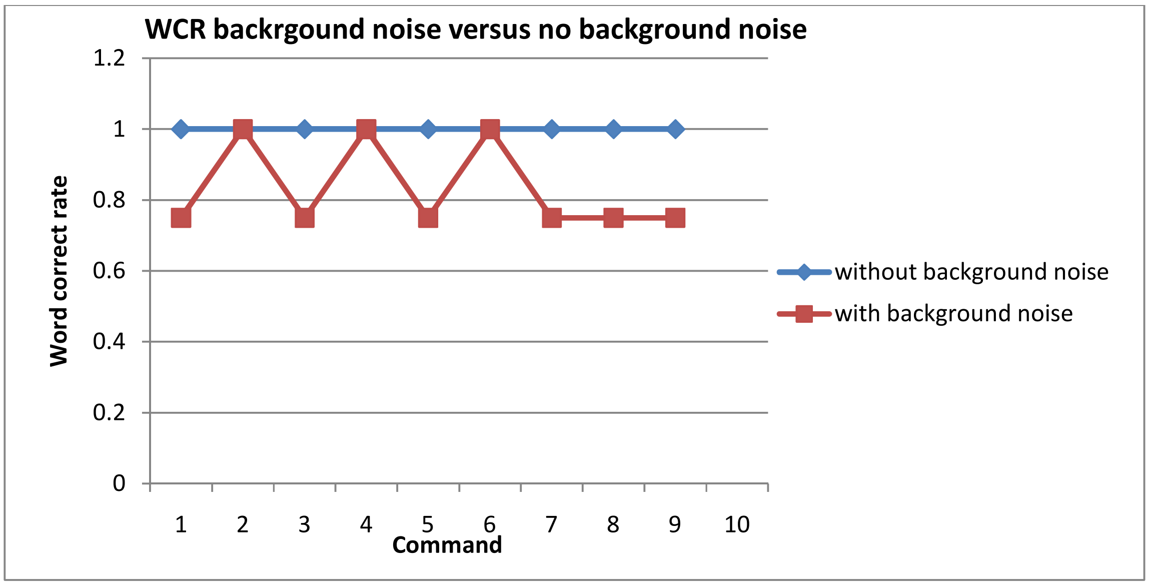


Figure 5.1 - Word Correct Rate using background noise variable

However this is not the case when running different possible audio commands without background noise. Table 5.2 shows that the WCR is not always 1. For specific commands, the rate decreases from 1 to 0.75 or 0.5. But still, the system is recognizing most of the words for distinct audio commands. We can notice here that the decrease of the WCR is due to the difference of words existing in the command. Some words are not well recognized by the system which explains the slight decrease on the WCR. For instance the command 7 corresponds to the voice command (*Robot one go back place*) is not well recognized by the speech engine due to the misunderstanding of the word *go back*. Although this word was not recognized, the system recognized other word instead. The word go back might be not recognized due to its number of characters. This word is not long. The system could be more comfortable to recognize more long words such as "*move back*". When the word is long enough, the system do not mismatch it to other word, within the dictionary of the speech recognizer.

Command	Word Correct Rate without background noise
1	1
2	1
3	1
4	1
5	0.75
6	1
7	0.75
8	0.5

Table 5.2 Word Correct Rate with different audio commands

We have also created 9 audio files containing natural voice commands using 2 speakers natural speech. Nine more audio waveform files have been created using 3 speakers natural speech command. In total, we have created 18 additional audio files to test the strength of the system against the variation of the number of speakers. For every 9 audio files, we have computed the Average Word Correct Rate. We have calculated this Average for every variation of the number of speakers. Table 5.3 shows the result of the average rate corresponding to the number of speakers.

AVRWCR	Number of speakers
1	1 speaker
0.78125	2 Speakers
0.6666667	3 Speakers

Table 5.3 Average word correct rate versus number of speakers

When using 2 speakers for the natural voice command, the AVRWCR decreased modestly to 0.78125. This rate explains that the system is still accurate even when feeding the system with 2 speakers' voice command. The corresponding average rate decreases when using 3 speakers. The system could not recognize all the words but the average is still above the 50%. These results are promising regarding the variation of speakers. We have plotted the results in the following figure 5.2.

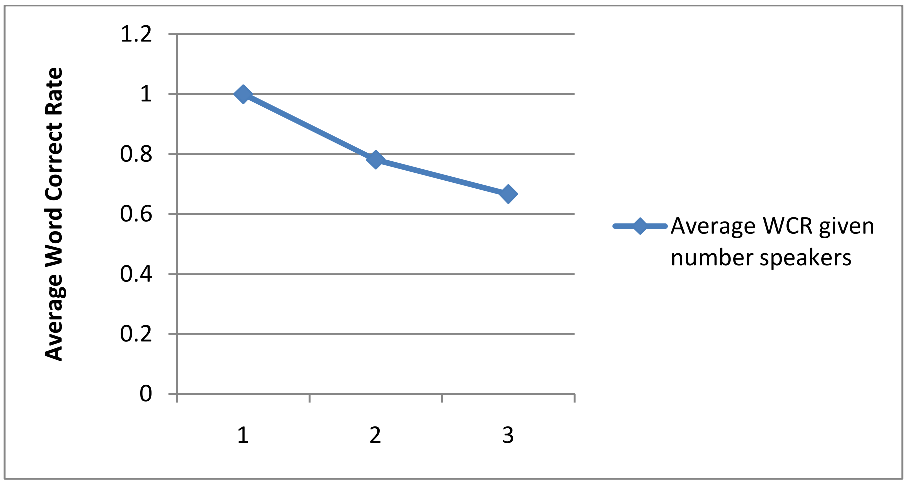


Figure 5.2 - Average word correct rate vs. - number of speakers and its trend

When varying the distance from the microphone and modifying the voice speed of the commands, the system has proved fair results. For that, we have created different audio files by first varying the distance from the microphone and by then adjusting the voice speed to accelerate the natural voice command. In order to adjust the speed, we have used the *wavePad Sound Editor.* We have first saved a natural voice command using the original speed adjustment which is 100%. We, then, augmented the speed adjustment by 50% and finally we doubled it to create other audio file with 300% speed adjustment. Table 5.4 and 5.5 shows the results of the word correct rate by varying both the distance and the voice speed of the audio commands.

Word Correct Rate	Distance
1	15cm
1	30cm
0.75	45cm

Table 5.4 Word correct rate versus Lateral distance

Word Correct Rate	Voice Speed
1	100% speed adjust
1	150% speed adjust
0.25	300% speed adjust

Table 5.5 Word correct rate versus voice speed

Results on table 5.4 show clearly that the word correct rate is slightly affected by the variation on the distance. The word correct rate decreased only when the speaker was 45cm far from the microphone.

The system has proved encouraging results when varying the speed of the voice command. But the word correct rate declined drastically when adjusting the speed to 300%. This makes sense since the speed of the voice command is really quick and therefore the system is unable to recognize many words. Distance and voice speed variation are exemplified in the following figures.

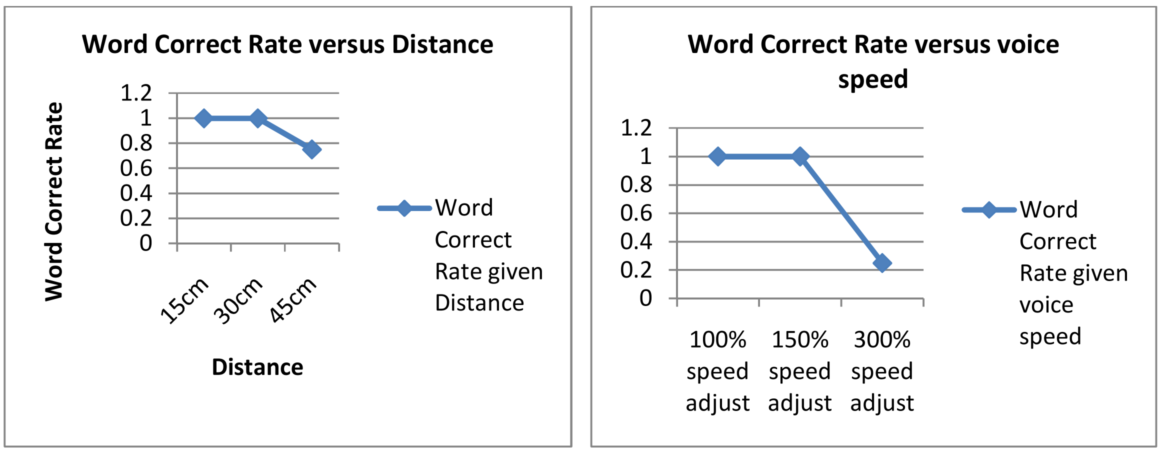


Figure 5.3 - Distance and voice speed variation

5.3.2 Success and False Action rate

Success and False Action rate are calculated against some of the variables to assess the robustness of the system as well as how our GUI and simulation are responding to different commands. More clearly, we have tested how well the robots are acting in the simulation phase based on different variables variations in the environment. Table 5.6 shows the results of both action and false action rate when feeding the system with different voice commands.

Command	Success Action Rate	False Action Rate
1	1	0
2	1	0
3	1	0
4	1	0
5	0	1
6	1	0
7	0	1
8	1	0

Table 5.6 Success and False action rate using different commands

The robots on the GUI are acting correctly to certain commands. This is demonstrated on the table by a success action rate of 1. But this is not the case for other voice commands. For instance, robots are acting differently to commands 5 (*robot 2 hand_me cup*) and 7(*robot 1 go back place*). In this case, the corresponding false action rate is 1. We can notice here that these results matches the results previously presented on table 4.2 corresponding to the calculation of the word correct rate against different voice commands. The word correct rate was 0.75 for both command 5 and 7. This confirms that the ASR system is not recognizing all the words in both commands 5 and 7. Hence, the value 1 of the false action rate is well explained. Despite the system is not recognizing some words in few commands, it behaves to these commands in a wrong way. Figure 5.4 depicts these results.

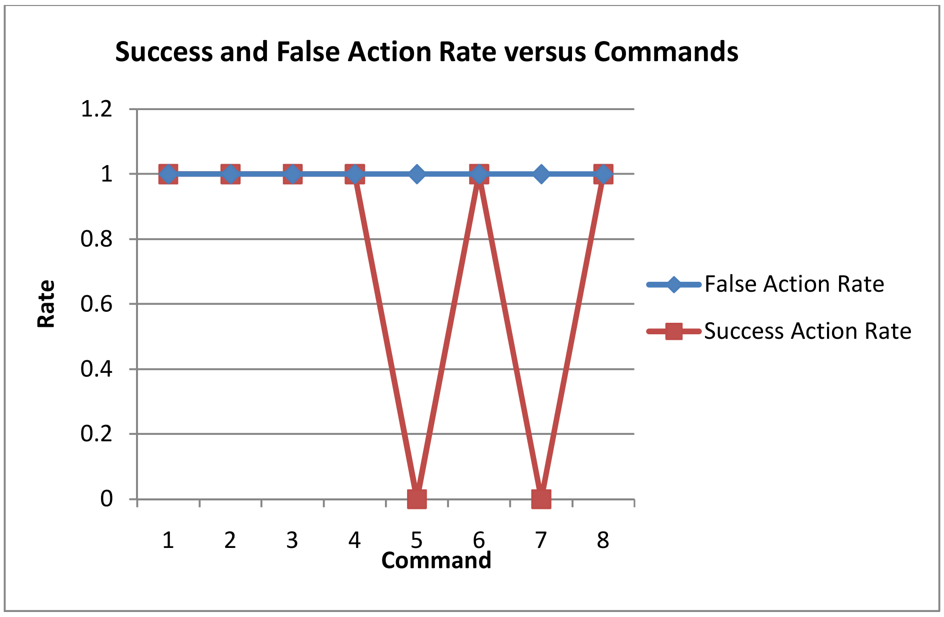


Figure 5.4 - Voice command variation

During the simulation phase of the system, the robots are acting correctly to the archived voice commands. In order to demonstrate how the system is behaving given different environment variables modification, we calculated the success and false action rate for both number of speakers and distance variation. We have created 9 audio files. For every 3 audio files, we varied the number of speakers within the same voice command. We have then calculated the average of the success action rate for every modification in the number of speakers. Table 5.7 illustrates these results. The success action rate decreases to 0.666 when adding other speaker to the voice command. The Average word correct rate has decreased as explained previously when adding other speaker. This explains the value of the success action rate. The system is not recognizing all the expected words from the voice command. For that reason, the robots are not acting to all the commands correctly.

Number of Speakers	False Action Rate	Success Action Rate
1	0	1
2	0.333	0.666
3	0.333	0.333

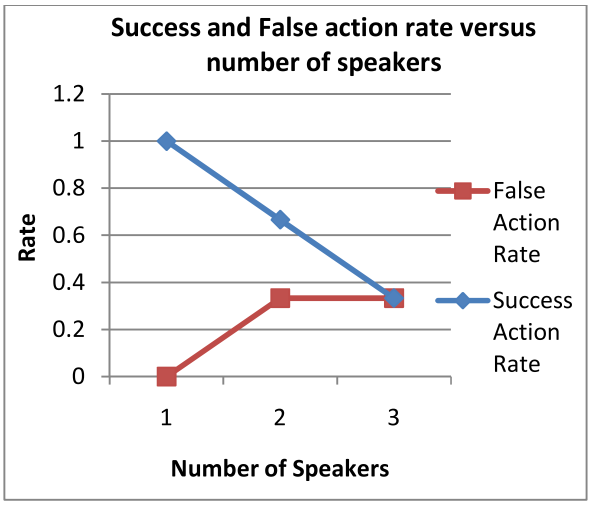
Table 5.7 Number of speakers variation

We have also computed the success and false action rate against the distance variation. Table 5.8 shows that the success action rate is not changing when being 15 and 30 cm far from the microphone.

Success Action Rate	False Action Rate	Lateral Distance from microphone
1	0	15
1	0	30
0	0	45

Table 5.8 Lateral distance variation

However both the success and false action rate are null when being 45 cm from the microphone. Referring to the results of table 4.4, the system is recognizing some words from the voice command since the rate obtained is 0.75.Although the system is recognizing many words, both success and false action rate are equal to 0. The system did not behave to the voice command. This might be a problem within the GUI simulation framework. We presented these measures in the following figure 5.5.



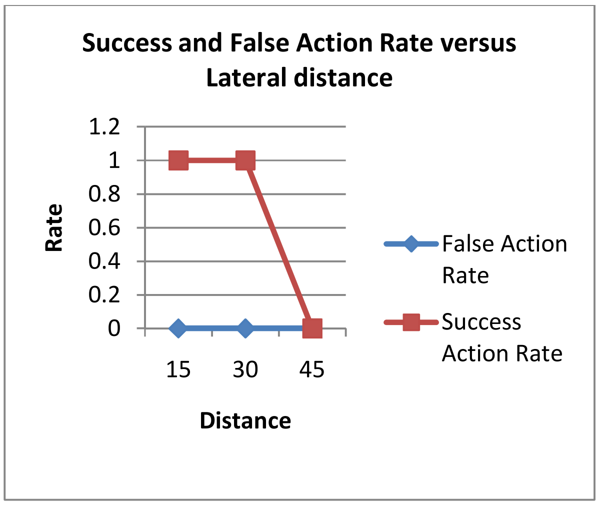


Figure 5.5 - Number of speakers and distance variations

5.3.3 Recognition time

We have also used the recognition time to assess the recognition speed of our system and how fast robots are behaving to the user natural commands. For that, we have calculated the recognition time with the same voice command run many times. In order to discuss recognition time results, we have also computed the time against the variation of the voice command. To verify how the recognition time is varying upon the variation of speakers' number, we have calculated the recognition time with different number of speakers within the natural voice command. For every number of speakers, we have used the same voice command but run many times to get the average recognition time. Results presented in the following tables confirm that, in average, the recognition time is few seconds. When using the same voice command, the recognition time was between 1 and 3 seconds. Generally speaking, the recognition time was good and has not exceeded 3 seconds. This is promising when using the system for humans to interact naturally with robots. We have also plotted the results obtained from previous tables.

Recognition Time (sec)	Iteration
2	1
2	2
3	3
1	4
1	5
2	6
1	7
2	8
2	9

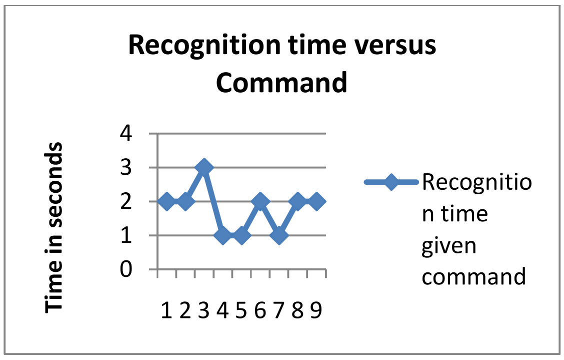
Table 5.9 Recognition time versus same command

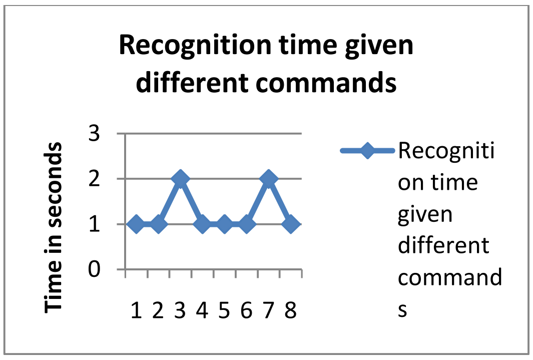
Recognition Time (sec)	Commands
1	1
1	2
2	3
1	4
1	5
1	6
2	7
1	8

Table 5.10 Recognition time versus different commands

Recognition Time	Number of Speakers
3.333	1
3.4	2
3.333	3

Table 5.11 Recognition time versus number of speaker





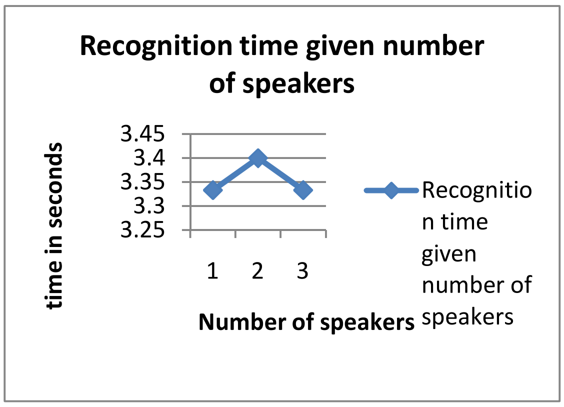


Figure 5.6 - Command and number of speakers' variation

5.4 Correlations

In order to resolve the relation between the evaluations measures and the variables, we have adopted the CORREL function within Excel. We have used this function to determine whether the evaluation measures and variables are related or not and if they are related, how strongly. The correlation coefficient extends from +1 to -1.When this coefficient is close to +1, it indicates a positive linear relationship between the two data sets. On the other side, a coefficient close to -1 indicates a negative linear relationship between them. The covariance of the data sets and the standard deviations of each data set are employed to calculate the correlation coefficient. In this section we are going to discuss the correlation between our evaluation measures and variables explained above.

- WCR and number of speakers

The obtained correlation coefficient is -0.9841108. This suggests that both the WCR and the number of speakers are strongly related negatively. More precisely, as the number of speakers increases, the word correct rate decreases. This confirms results already obtained since the system is not recognizing all the words while adding new speakers to the voice command.

- WCR and Distance Variation

The matching correlation coefficient is -0.866025404. These two data sets are related negatively. The WCR is not strongly related to the distance variation since the coefficient is not very close to -1. This confirms the results presented in table 5.4. In fact the WCR did not change when the speaker is 30cm far from the microphone.

- WCR and Voice speed variation

The calculated correlation coefficient is -0.970725343. The word correct rate and the voice speed variation variable are negatively related. This is demonstrated by the results showed in table 5.5. The word correct rate decreased to 0.25 while the voice speed increased to 300%.

- Success action rate and number of speakers

The corresponding correlation coefficient is -0.999999625. The success action rate and the number of speaker variable are negatively strongly related. The value is almost -1. When the number of speakers goes up, the success action rate goes down. This confirms results on table 5.7.

- False action rate and number of speakers

We have obtained 0.866025404 as a correlation coefficient between the two variables. The false action rate is positively related to the number of speakers' variable since the value is close to +1. The false action rate increased when the number of speakers goes up.

- Success action rate and distance variable

The correlation value obtained between the action success rate and the distance variable is -0.866025404. The two samples are negatively related. However, they are not strongly related and this is depicted by results on table 5.8.

- Recognition time and number of speakers

The corresponding correlation coefficient is 0. This indicates that there is no correlation between the recognition time and the number of speakers. More clearly, the recognition time is not really affected by the number of speakers' variations.

5.5 Summary

This chapter discussed the experiments conducted in order to quantitatively evaluate the performance of the proposed system. The proposed natural speech understanding system showed encouraging results especially in the behaving of robots within the GUI simulation. However, the system still needs improvement in terms of recognizing many natural voice commands. The proposed system has been trained with only 67 audio files. Better performance can be obtained by expanding the grammar and by adding more vocabulary words to the dictionary to be recognized.

Chapter 6

Conclusion and Future work

The main objective of this research work was to implement a new system that provides an enhanced speech recognition framework. This system was also used as a testbed to enable human-robot verbal interaction. The proposed approach consisted of applying a design pattern-based user interface and a user-independent automatic speech recognizer with a modified grammar module in the context of human-robot interaction.

We described in chapter 3 the adopted components and methods to implement our system. The modified Julius speech recognizer joined with the Java application module has shown promising results regarding the recognition accuracy and robustness of the system under different working conditions. Discussed results in chapter4 have demonstrated that the proposed system performs well when simulating robots' actions on the graphical user interface. Robots are acting accurately to many voice commands. In addition, robots were able to handle more complex commands and to go beyond recognizing simple stand-alone commands. Analysis of results in chapter 4 of the word correct rate, the recognition time and the success / false action rates we introduced proved that in terms of speech variations, the rate was overall good. The speech input was modified and stretched to assess the robustness of the system toward more complex voice commands.

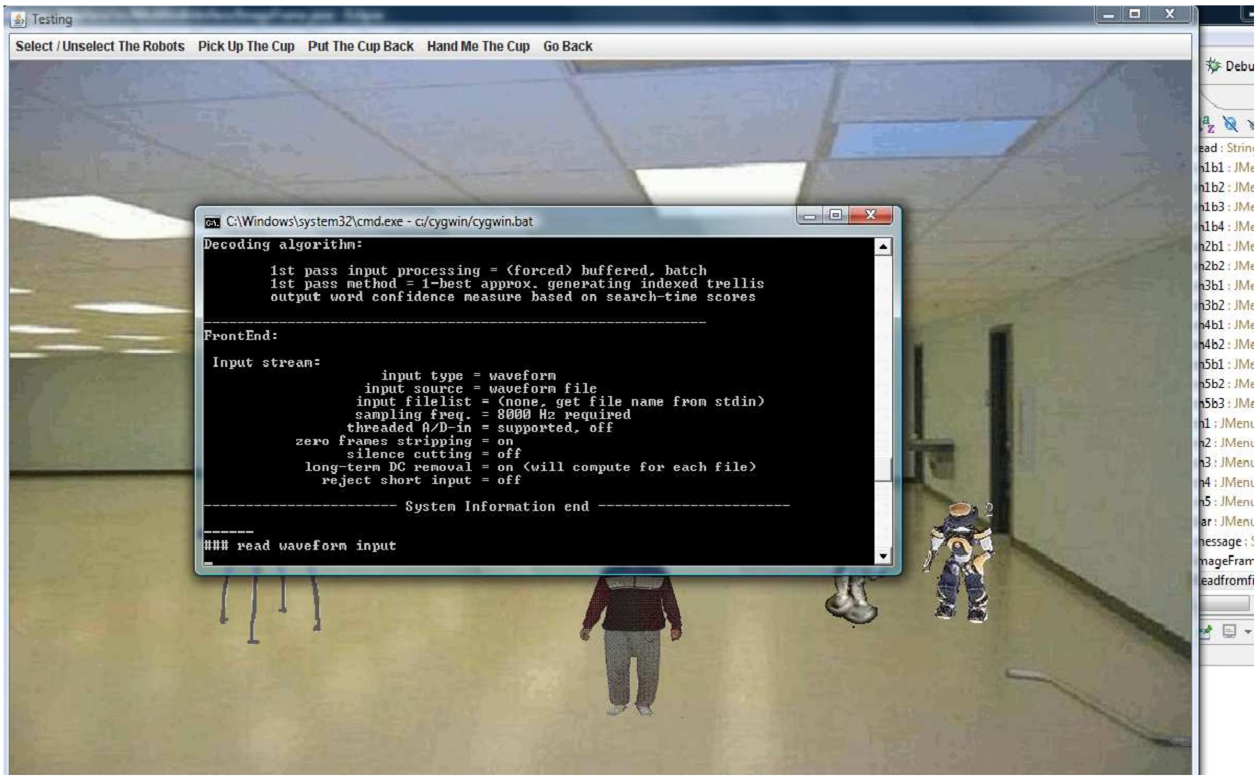
The obtained results are encouraging and we can see a lot of potential in this system toward an enhanced speech recognition system and a testbed enabling advanced human-robot verbal interaction. This shows that our system is promising, and with further enhancements and development it could lead to high accurate human-robot verbal interaction system.

Further improvements can be applied to the system. The system can be enhanced to recognize diverse natural voice commands of variable length. Hence, instead of training the proposed system with only 67 audio files, better performance can be obtained by expanding the grammar, adding more vocabulary words to the dictionary to be recognized and by training the system with much more data. As a future work, we consider porting the proposed system to the PeopleBot robotic platform currently used in the PAMI lab in order to study cognitive robotics. The current system can be set to real PeopleBot robots to test the platform in real world environment. The ported system in PeopleBot robots can be used in further research works and in the area of cognitive robotics. These robots, using an enhanced natural speech understanding framework, can later be employed in assistive robotic systems in order to assist elderly people in several tasks.

Appendix A User Guide

This appendix provides a detailed user guide explaining how to use the client tool for our human-robot verbal interaction based on speech recognition framework. Throughout this guide, detailed screen shots will be provided showing the simulation of robots and their actions to some human voice commands.

Figure A.1 presents how the speech engine is started from the client application. As a first step, the user executes the java application which runs automatically the modified Julius engine on background. The altered Julius engine is ready to recognize voice wave files archived after recording human voice commands.

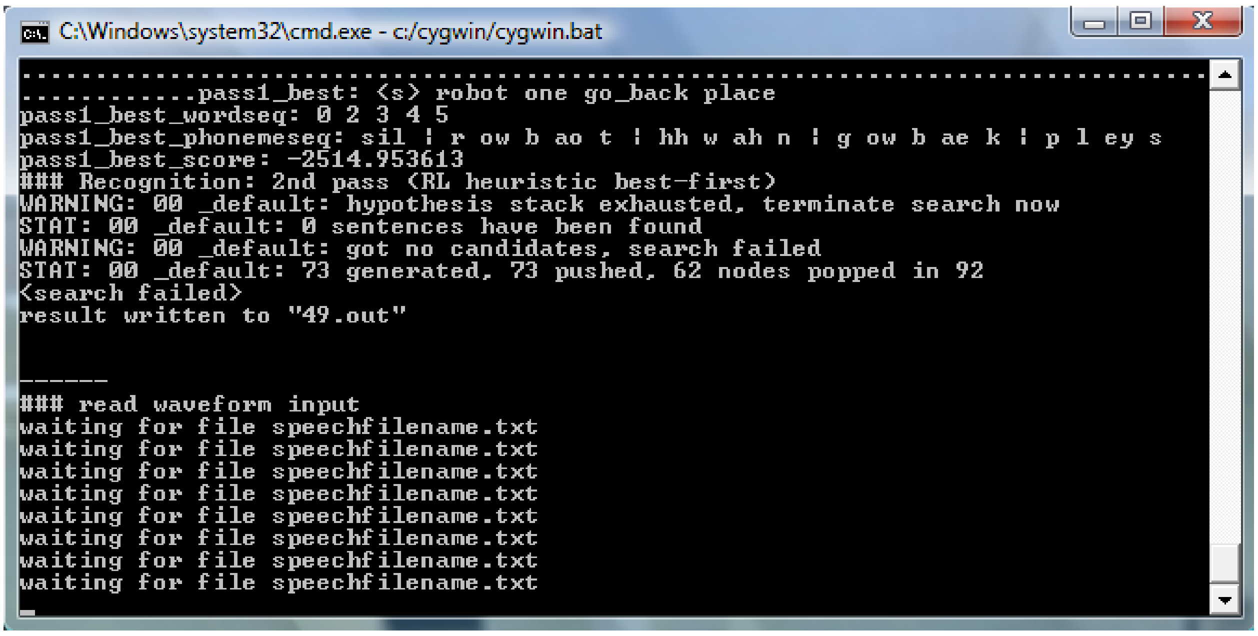

Testing
Select / Unselect The Robots
Pick Up The Cup
Put The Cup Back
Hand Me The Cup
Go Back
C:\Windows\system32\cmd.exe - c:/cygwin/cygwin.bat

```
Decoding algorithm:

        1st pass input processing = (forced) buffered, batch
        1st pass method = 1-best approx. generating indexed trellis
        output word confidence measure based on search-time scores

------------------------------------------------------------
FrontEnd:

 Input stream:
                     input type = waveform
                   input source = waveform file
                  input filelist = (none, get file name from stdin)
                  sampling freq. = 8000 Hz required
                 threaded A/D-in = supported, off
           zero frames stripping = on
                 silence cutting = off
            long-term DC removal = on (will compute for each file)
              reject short input = off

----------------------- System Information end -----------------------

------
### read waveform input
_
```

Figure A.1 - Application run and speech engine startup

After running the Java application, the speech engine will be waiting to read the voice file within the speechfilename.txt. As explained previously, the speech recognizer was modified to wait for voice archived files. Voice commands are archived in voice wave file. The name of the saved wave file is located in the speechfilename.txt which the recognizer will use to recognize user voice commands.

The following Figure illustrates the speech recognizer engine waiting to recognize voice files.



C:\Windows\system32\cmd.exe - c:/cygwin/cygwin.bat

```
..........................................................................................
.............pass1_best: <s> robot one go_back place
pass1_best_wordseq: 0 2 3 4 5
pass1_best_phonemeseq: sil | r ow b ao t | hh w ah n | g ow b ae k | p l ey s
pass1_best_score: -2514.953613
### Recognition: 2nd pass (RL heuristic best-first)
WARNING: 00 _default: hypothesis stack exhausted, terminate search now
STAT: 00 _default: 0 sentences have been found
WARNING: 00 _default: got no candidates, search failed
STAT: 00 _default: 73 generated, 73 pushed, 62 nodes popped in 92
<search failed>
result written to "49.out"


------
### read waveform input
waiting for file speechfilename.txt
waiting for file speechfilename.txt
waiting for file speechfilename.txt
waiting for file speechfilename.txt
waiting for file speechfilename.txt
waiting for file speechfilename.txt
waiting for file speechfilename.txt
waiting for file speechfilename.txt
```

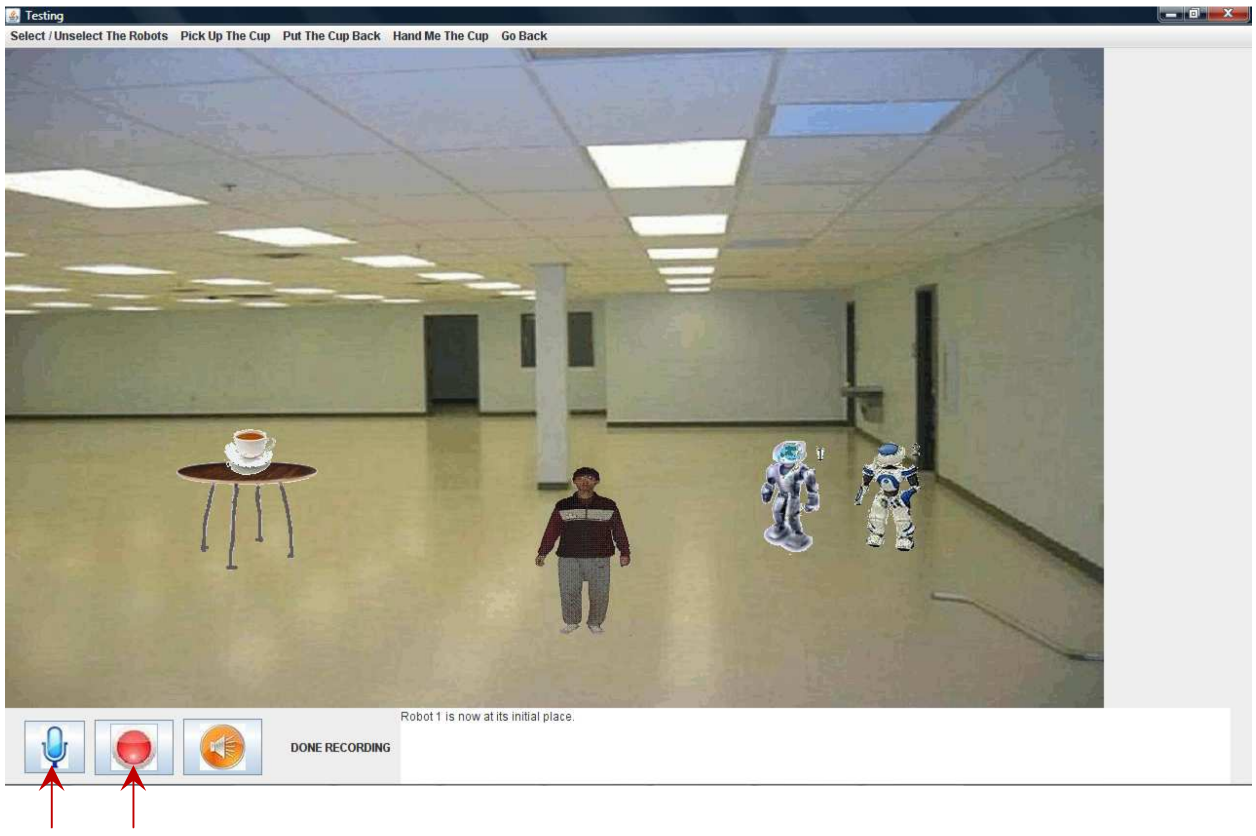
Figure A.2 - speech recognizer waiting for voice input

The implemented framework works as follows:

- The user records voice input using the Java graphical user interface: The speech engine is running on background and waiting to recognize speech input.
- After recording the voice input, the user stops the recording after pushing the *Stop* button within the User interface.

- When the user stops recording, the speech input is saved in voice.wav file and then the name of the file is automatically located in the speechfilename.txt file
- The speech recognizer will recognize the speech and generate the matching text into the screen.
- After recognizing the .wav file, the speech engine also creates an output file “voice.out” containing information about the recognized speech as well as the text corresponding to the voice command.
- This text will then be parsed and used to act on the robots within the interface.
- Robots will act given voice commands and their behavior will be simulated on the interface.

The Following screen shots illustrate the detailed process and shows some of the robots tasks.



Robot 1 is now at its initial place.

Figure A.3 - Global graphical user interface

Figure A.4 shows how the speech recognizer parses the speech and prints the output text on the command screen.

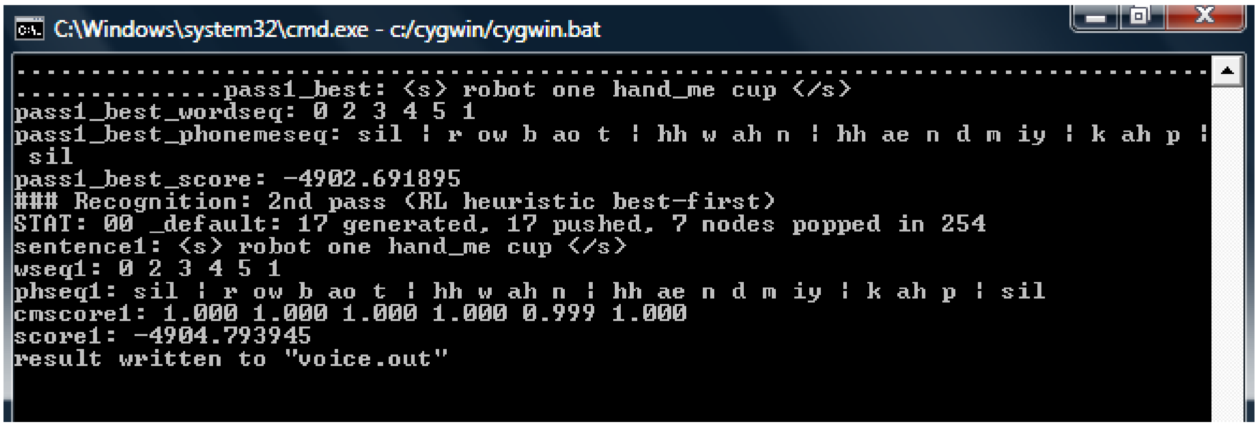
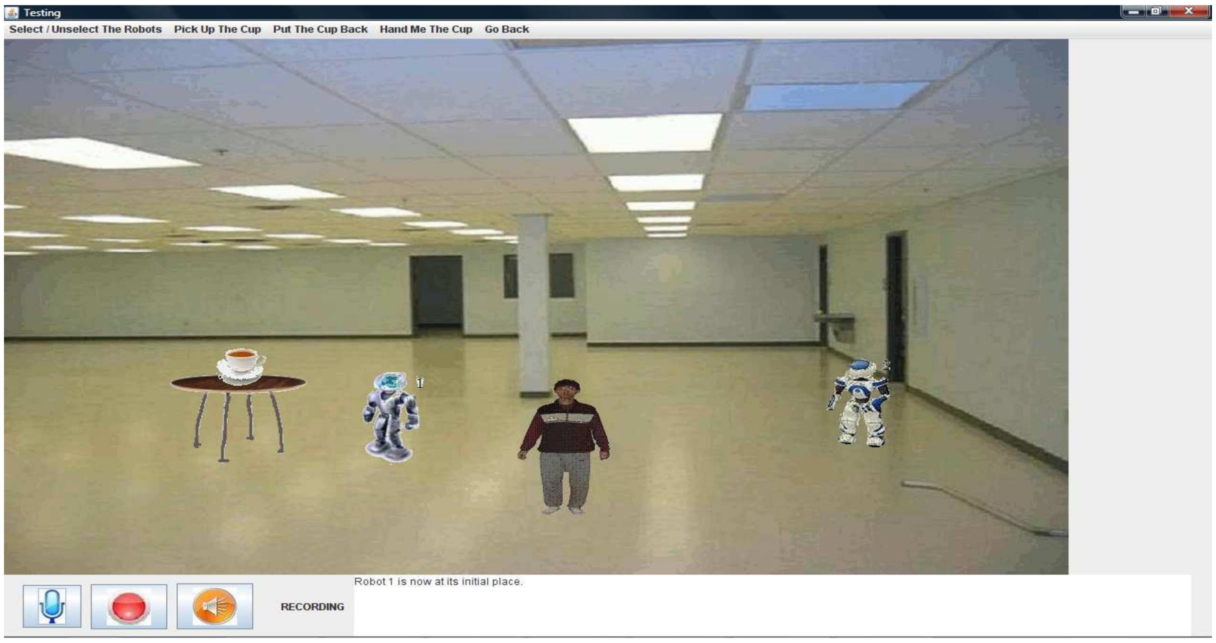

```
C:\Windows\system32\cmd.exe - c:/cygwin/cygwin.bat
..........................................................................................
..............pass1_best: <s> robot one hand_me cup </s>
pass1_best_wordseq: 0 2 3 4 5 1
pass1_best_phonemeseq: sil | r ow b ao t | hh w ah n | hh ae n d m iy | k ah p |
 sil
pass1_best_score: -4902.691895
### Recognition: 2nd pass (RL heuristic best-first)
STAT: 00 _default: 17 generated, 17 pushed, 7 nodes popped in 254
sentence1: <s> robot one hand_me cup </s>
wseq1: 0 2 3 4 5 1
phseq1: sil | r ow b ao t | hh w ah n | hh ae n d m iy | k ah p | sil
cmscore1: 1.000 1.000 1.000 1.000 0.999 1.000
score1: -4904.793945
result written to "voice.out"
```

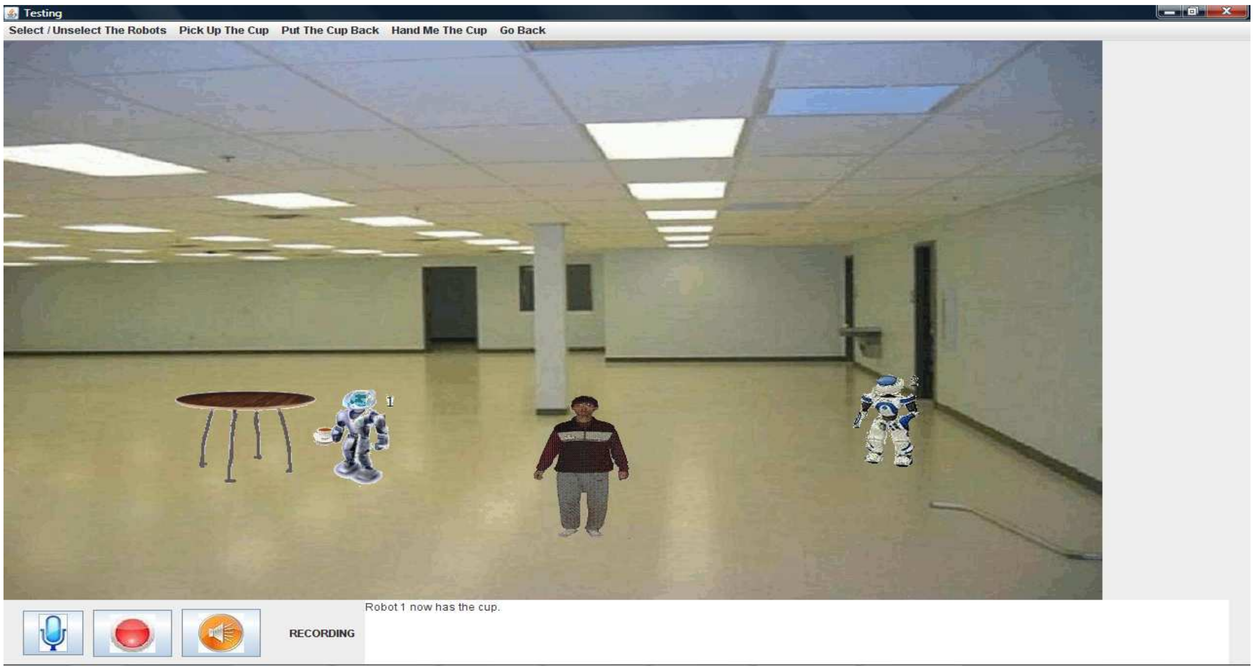
Figure A.4 - Recognition of the voice command

The robots then acts on the user voice command input as shown in the following screen shots.



Robot 1 is now at its initial place.

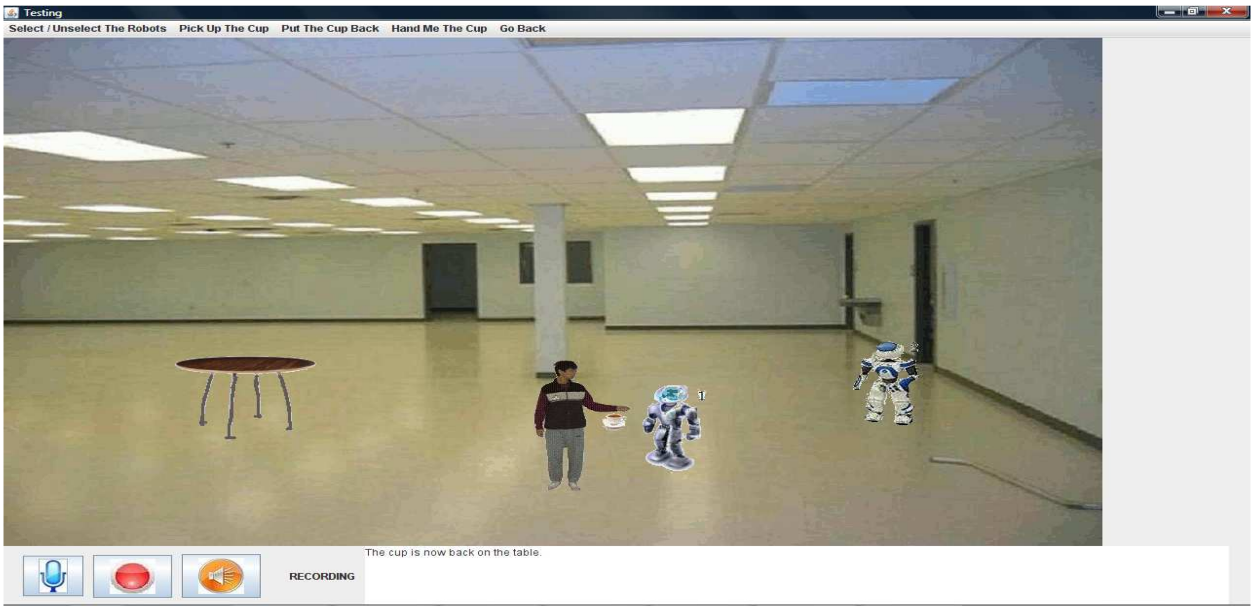
Figure A.5 – Going to table



Robot 1 now has the cup.

Figure A.6 – Picking up the cup

The following screen shot depicts another action of the robot to the voice command *“Robot 1 hand me cup”*



The cup is now back on the table.

Figure A.7 – Handing over the cup

Bibliography

[1] S. Sabanovic, M.P. Michalowski, and R. Simmons, "Robots in the wild: observing human-robot social interaction outside the lab," in *Proceedings of the 9th International Workshop on Advanced Motion Control (AMC 2006)*, March, 2006, pp. 596-601.

[2] A. Khamis, M. Kamel, and M.A. Salichs, "Human-Robot Interfaces for Social Interaction," *International Journal of Robotics and Automation*, vol. 22, 2007.

[3] R. Brueckmann, A. Sheidig, and H.M. Gross, "Adaptive Noise Reduction and Voice Activity Detection for improved Verbal Human-Robot Interaction using Binaural Data," in *IEEE International Conference on Robotics and Automation*, Roma,Italy, 2007, pp. 1782-1787.

[4] N. Esau, L. Kleinjohann, B. Kleinjohann and A. Austermann, "Fuzzy Emotion Recognition in Natural Speech Dialogue," in *IEEE International Workshop on Robots and Human Interactive Communication*, 2005, pp. 317-322.

[5] K. Scherer, "Vocal communication of emotion," in *Speech Communication*, Elsevier, 2003, pp. 227-256.

[6] F. Karray, O. Basir, M. Kamel, and J. Sun, "Fuzzy Logic-Based Natural Language Understanding and Its Application to Speech Recognition," Electrical and Computer Engineering Department, University of Waterloo, Ontario, Canada.

[7] Comrie, "Language Universals and Linguistic Typology", Oxford, Blackwell, 1989.

[8] D. Bates, M. Brown, M.. Fisher, W. Hunicke-Smith, K. Pallett, D. Pao, C. Rudnicky, A. Shriberg and E. Dahl, "Expanding the scope of the ATIS task: the ATIS-3 corpus," in *ARPA Workshop on Human Language Technology*, 1994, pp. 43-48.

[9] S.B. Shouraki, R. Halavati, and M. Alemzadeh, "Fuzzy Classification by Multi-layer Averaging An application in speech recognition," in *3rd Internati onal Conference on Informatics in Control, Automation and Roboti cs,* Setubal, Portugal, August 2006.

[10] S.B. Shouraki, M. Eshraghi, M. Alemzadeh, P. Ziaie, and R. Halavati, "A Novel Fuzzy Approach to Speech Recognition," on *Fourth International Conference Hybrid Intelligent Systems, HIS '04,* Tehran, Iran, 2004.

[11] J Holland, *Adaptation in Natural and Artificial System*, Bradford Books, 1992.

[12] M. Majewski and W. Kacalak, "Intelligent System for Natural Language Processing," in *Springer-Verlag Berlin Heidelberg*, Poland, 2006, pp. 742-747.

[13] M. Kacalak and W. Majewski, "Intelligent Layer of Human-Machine Voice Communication," in *The 9th International Conference KES2005,* Melbourne, Australia, 2005.

[14] J.A. Starzyk and F.A. Elmisery, "A neural network based on sequence learning for speech recognition," in *The International Conference on Computer Engineering & Systems ICCES 2008,* Cairo, 2008, pp. 139-142.

[15] M.A. Arbib, "The handbook of brain theory and neural networks," , Cambridge, MA.

[16] S.C. Woo, A.S. Loh, R. Osman, and C.P. Lim, "Speech Recognition Using Artificial Neural Networks," in *The First International Conference on Web Information Systems Engineering,* Hong Kong, China, 2000, pp. 419-423.

[17] L. Daubechies, "Orthogonal bases of compactly supported wavelets ," in *Communication on Pure and Applied Mathematics,* pp. 909-996.

[18] F. Karray, A. Abghari, S. Podder, and S. Ullah, "Soft Computing-based Approach for Natural Language Call Routing Systems," in *The 9th International Symposium on Signal Processing and Its Applications ISSPA 2007,* Sharjah, 2007, pp. 1-4.

[19] F. Karray and C. De Silva, *Soft Computing and Itelligent System Design*, Addison Wesely Publishing, 2004.

[20] T. Kohonen, *Self-Organization and Associative Memory*, 2nd ed. Berlin, Heidberg, New York: Springer, 1988.

[21] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey, *Human-Computer Interaction*, first ed.: Addison Wesley, 1994.

[22] D.R. Olsen and M.A. Goodrich, "Metrics for Evaluating Human-Robot Interactions", in *Proceedings of PERMIS,* Utah, 2003.

[23] J. Casper, "Workflow study on human-robot interaction in USAR," in *Proccedings of ICRA,* 2002, pp. 1997-2003.

[24] K. Haigh and H.K. Yanco, "Automation as caregiver: a survey of issues and technologies," in

Proccedings of the AAAI-2002 on Automation as Caregiver : The Role of Intelligent Technology in Elder Care, Edmonton, Alberta, 2002.

[25] H.A. Yanco, *A Robotic Wheelchair System: Indoor Navigation and User Interface,* Springer-Verlag, 1998.

[26] M. Lansdale and T. Ormerod, *Understanding interfaces: A handbook of human-computer dialogue*. Landon: Academic Press, 1994.

[27] D. Goren-Bar, "Designing model-based intelligent design systems," in *In information Modeling in the New Millennium Idea Group* , London, 2001.

[28] A. Abella and A. Gorin, "Construct Algebra: Analytical Dialog Management," in *In Proccedings Annual meeting of the ACL*, 1999.

[29] D.R. Olsen and M.A. Goodrich, "Metrics for Evaluating Human-Robot Interactions," in *ACM/IEEE International Conference on Human-Robot Interaction*, Utah, USA, 2006.

[30] M. Rauterberg and P. Steiger, "Pattern Recognition As a key Technology For the Next Generation of User Interfaces," in *In Proceedings of SMC'96, 4, IEEE Catalog Number: 96CH35929*, 1996, pp. 2805-2810.

[31] O. Hiroshi, N. Kazuhiro, T. Lourens, and K. Hiroaki, "Human-Robot Interaction Through Real-Time Auditory and Visual Multiple-Talker Tracking," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii, 2001, pp. 3-29.

[32] J. Schmid, T. Kollar, E. Meisner, V. Sweetser, D. Safer, and C. Brown, "Mabel: Building a Robot Designed for Human Interaction," in *The Mobile Robot Competition and Exhibition, TR WS-02-18*, 2002, pp. 24-32.

[33] R. Stiefelhagen, C. Fügen, P. Gieselmann, H. Holzapfel, K. Nickel, and A. Waibel, "Natural Human-Robot Interaction using Speech, Gaze and Gestures," in *IROS 2004, IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[34] W. Burgard, A.B. Cremers, D. Fox, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun , "The interactive museum tour-guide robot" , in *In Proccedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, Wi, 1998.

[35] L. Horswill, "A vision-based artifical agent", in *In the Proceedings of the Eleventh National Conference on Artifical Intelligence* , Japan, 1993.

[36] G. Antoniol, "Experiencing real-life interaction with the experimental paltform of MAIA," in *In*

proceedings of the 1st European on Human Comfort and Security, 1994.

[37] I. Androutsopulos, "A prinicipled Framework for Constructing Natural language Interfaces to Temporal Databases," University of Edinburgh, PHD thesis 1996.

[38] H. Asoh, T. Matsui, J. Fry, J. Asano, and S. Hayamizu, "A spoken dilaogue system for mobile office robot," in *Proceedings of Eurospeech 99*, Budapest, 1999, pp. 1139-1142.

[39] T. Matsui, H. Asoh, J. Fry, Y. Motomura, F. Asano, T. Kurita, I. Hara, and N. Otsu, "Integrated natural spoken dialogue system of Jijo-2 mobile robot for office services," in *In the proceedings of the AAAI-99*, 1999.

[40] L. Versweyyveld, "Voice-controlled surgical robot ready to assist in minimally invasive heart surgery," New Orleans, 1998.

[41] Microsoft Corporation, "http://www.microsoft.com/speech/downloads/sdk51/", Last accessed March 5, 2010.

[42] V. Kulyukin, C. Gharpure, P. Sute, N.D. Graw, and John Nicholson, "A Robotic Wayfinding System For the Visually Impaired," in *Proceedings of the Innovative Applications of Artificial Intelligence (IAAI)* , San Jose, California, 2004, pp. 864-869.

[43] V.A. Kulyukin, "On Natural Language Dialogue with Assistive Robots," in *The ACM/IEEE International Conference on Human-Robot Interaction*, Utah, USA, 2006, pp. 164-171.

[44] A. Lee, T. Kawahara, and K. Shikano, "Julius an Open Source Real-Time Large Vocabulary Recognition Engine," in *The 7th European Conference on speech communication and Technology,* Denmark, 2001.

[45] "www.VoxForge.com", Last accessed March 5, 2010.

[46] V. Ramachandran, "Design Patterns for Building Flexible and Maintainable J2EE Applications," January 2002.

[47] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide* , Addison-Wesley edition, 1999.

[48] I. McCowan, D. Moore, J. Dines, D. Gatica, M. Flynn, P. Wellner, H. Bourlard, "On the Use of Information Retrieval Measure For Speech Recognition Evaluation," IDIAP Research Institute, Switzerland, Research 2005.

[49] Z. Maamar and Q.H. Mahmoud, "Applying the MVC Desin Pattern to Multi-Agents Systems," University of Guelph,Guelph, University of Dubai, Canada , UAE.