7

User Voice Assistance Tool for Teleoperation

Oscar Reinoso, César Fernández, and Ramón Ñeco

Universidad Miguel Hernández
Dpto. Ingeniería de Sistemas Industriales
Avda. Universidad. Edif. Torreblanca. 03202 Elche-Alicante, Spain
{o.reinoso,c.fernandez,ramon.neco}@umh.es

Summary. A teleoperation tool that allows to interact with the remote environment in a more comfortable and flexible way is presented in this chapter. Working over a classical teleoperation environment, the goal is to reach a higher level of abstraction in the user commands. The tool allows the operator to interact with the remote environment through natural language recognition. This system is able to interpret and execute the commands formulated by the operator in natural language, according to the elements present in the remote environment. An error feedback module has been designed in order to take into account the on-line correcting information expressed by the operator during the execution of a task in the remote environment. The proposed voice assistance tool has been designed as a module in a novel teleoperation architecture, which allows to integrate multiple assistance tools.

7.1 Introduction

Teleoperation involves cooperation between the operations made by the user in a local environment and the actions exerted by the physical components in the remote environment. In a robotic teleoperated system the operator acts over a robot and the elements in the remote environment to perform the appropriate task [1]. Usually, teleoperation has been used in dangerous or inaccessible environments for the operator such as nuclear industry, reactor maintenance, spatial activities or dismantling tasks [1, 2]. However, the number of teleoperated systems has been significantly increased in the last years due to the possibilities offered by the World Wide Web [3] and the new tools that offer new possibilities to manage the remote environment.

In occasions, the environment is poorly known, and the conditions of perception are difficult. The operation reliability depends thus on all the data sensed around the work area and the way they are reproduced for the operator [2]. Classical telerobotic systems are usually composed of a vision system and a force sensor in the remote environment and a master arm in the local environment in such a way that the operator can manage the robot based on the information reported by these sensors.

However, some of the tasks made by the operator in a teleoperated way require a need of intelligent automation. In teleoperation, this intelligent automation

M. Ferre et al. (Eds.): Advances in Telerobotics, STAR 31, pp. 107–120, 2007.
springerlink.com

means that the tasks are carried out automatically with a supervisory control by human operators. In this way it is extremely important the assistance to operator's decision. In this chapter, a voice assistance tool that allows to interact with the remote environment in a more comfortable and flexible way is presented.

Natural language programming offers a great possibility to the operator to easily control and manage the remote environment [4, 5, 6]. Voice processing allows to establish a natural dialogue between a human and the robot in the remote environment in such a way that a not-qualified user of the system can manage the robot through a semantics that represents the environment and its relationship with the robot. The tool described in this chapter allows the operator to interact with the remote environment through natural language recognition.

We should mention that, of course, natural language interfaces are not adequate for all types of robotic commands. There are languages or human-friendly interfaces which are not natural but which are better for particular applications. Specifically, in the applications tested using the tool presented in this chapter, it is shown that natural language input can be an efficient technique for high level commands (HLC) and that for low level commands (LLC) [1] other input interfaces less natural are more appropriate.

7.2 Natural Language Interface

One of the simplest, most natural and highly user acceptable interfaces between human and teleoperated robots is surely one where the human operator speaks to the robot. This is the main motivation of the review and work presented in this chapter. One of the main advantages of voice recognition is that a single word or a simple sentence can communicate a complete procedure or a complex data structure. This motivation is confirmed with the fact that in recent years a growing interest exists in robots that are designed specifically to interact with operators who are not roboticists, such that they can respond to commands with voice. In order to be really useful in domestic and industrial environments, the robot should be able to interact in an appropriate way with its environment and with its human operators.

This section describes generalities about a module of natural language processing to teleoperate a robot, and section 7.3 presents a feedback processing and error correction in a real-time robotic teleoperated system. The overall robot control system consists of the following modules: speech recognition module, bilateral control module, vision module, learning module, and the robot controller and interface. The speech recognition module includes a natural processing system that permits understanding of high level natural language commands, and provides a low level program, directly executable by the robot.

Human-robot voice interaction is different from human-computer interaction mainly due to the manipulative power of the robot in a physical environment. The robot has a physical configuration and can operate in the environment during dialogue or as a result of a voice command. However, research on human-computer dialogue management is a good foundation for human-robot dialogue

management [7]. In the translation process of a command expressed in natural language to an executable program for the robot, the following characteristics have to be taken into consideration:

- There is no a target language universally-accepted for the high level tasks usually performed by a robot. This means that an appropriate *intermediate language* should be designed [4].
- The robot should have access to static geometric information about its working environment, as well as dynamic information, especially in interfaces for real-time systems.
- The necessary information to process the commands is not as structured as the information needed for other speech interfaces. Therefore, a general *knowledge representation* mechanism should be devised, including all the necessary aspects (information about the context, geometry, the robot's activities, sensor readings, etc.). At the same time, this general mechanism should be adaptive to specific robotic systems.
- The expressions that an operator can speak by using an interface to a robot are very heterogeneous. The translation process is, therefore, more complex since it needs more semantic information.

7.2.1 Related Works

This section describes some works directly related to the topic presented in this chapter. For a more general review (speech recognition details, speech generation) see [8].

Automatic speech recognition and natural language processing have been used as a powerful tool for human-computer communication [9, 10]. Some researchers have developed and demonstrated robots with a natural spoken language-based interface in a limited and technically restricted framework. The first complete system was SAM (Speech Activated Manipulator), where the authors implemented a discrete speech recognition system to control a robotic arm [6]. Crangle and Suppes [5, 11] devised a theoretical-oriented system to program a mobile robot and a robotic arm. Nevertheless, this system was not linked to a specific speech recognition module. They also gave a first adaptation technique applied to distinguish the different executions of the same verb to different execution circumstances or to different operators, focusing on the specific application of control devices for the disabled.

In the work described in [12] a formal model was designed in order to represent computationally the intentions of the user in dialogue systems. In [13] the authors incorporate a discrete speech recognition system (with some natural language capabilities) to a teleoperated robot which was applied to electrical maintenance tasks. In [14], the author describes new language-processing methods suitable for human-robot interfaces. These methods enable a robot to learn linguistic knowledge from scratch in unsupervised ways, at a low level (speech) processing through statistical optimization. In [15] the author describes a successful natural language interface to a mobile robot working in an office environment.

More recently, in works as [16,17], integration of natural language and gesture understanding was made in order to obtain a more natural interface in space and medical robotic applications. One of the most sophisticated and integrated systems can be found in [18]. It is a communicative humanoid robot with a hand and graphic face, which appears on a small monitor in front of the user. It is capable to perform face-to-face dialog, in real time, with a human user with various hand gestures, facial expressions, body language and meaningful utterances, which can be used to guide the humanoid-like robot.

7.2.2 Language Analysis and Understanding

In order to process natural language, we need to combine our understanding of small textual units to understand larger ones. The main objective of natural language processing theory is to show how these larger units of meaning arise out of the combination of the smaller ones, which is modeled by means of a *grammar*. It is traditional to divide the processing task into syntax and semantics, where syntax describes how the different formal elements of a sentence can be combined and semantics describes how the interpretation is computed. The grammar can be thought as the encoded linguistic knowledge, which is 'static' and separated from the processing components (the analysis algorithms). Basically, the grammar consists of a lexicon (a database of words or groups of words) and rules that syntactically and semantically combine words and phrases into larger phrases and sentences.

More specifically, in natural language processing we can distinguish several processing phases which coincide with the distinct steps in the process of understanding a natural-language command. These phases are the following. (1) Phonological and morphological processing; (2) Syntactic analysis; (3) Semantic analysis; and (4) Pragmatic analysis. Very briefly, each one of these phases is understood as follows.

The phonological and morphological processing is the processing of phonemes into basic units called morphemes, and then the processing of these morphemes into words. The syntactic analysis is the analysis of the order in which words are combined to form commands and, as said before, the syntactic knowledge is represented using formal grammars. The semantic analysis is the phase in which the system obtains the meaning of the individual words and how the whole meaning of a command is built up from the meanings of the words used in it. In the analysis of a verbal command to a teleoperated robot the meanings of the words are all the information we need to make the robot execute the command (including the parameters and the translation into the robot programming language). Pragmatic analysis is the phase in which the system process everything else that affects the use and interpretation of the natural language command in a specific context or situation. In a robotic application, the system uses the pragmatic analysis to find the values of some parameters or to make processing decisions that could not be made just using syntactic or semantic information.

As said before, a grammar describes the sentences that make up a language. It contains a finite number of rules that specify which sentences belong to the language and, at the same time, what is their syntactic structure. This way, we can obtain the underlying structure of the commands to the robot and extract its translation into the robot language. In the tool presented in this chapter we use an extended form of the classical *phrase-structure grammar*, which is defined as a tuple $G = (V, V_T, V_{NT}, P, C)$, where:

- V_T is the finite set of *terminal symbols* which correspond with the words or symbols that can appear in a command.
- V_{NT} is the finite set of *nonterminal symbols* which correspond to grammatical categories in which the language has been structured.
- $V = V_T \cup V_{NT}$ is the set of all the symbols in the grammar.
- P is the set of production rules of the form $p : a \rightarrow b$, which can be read as: from a we may derive b, where a is a combination of nonterminal symbols and b is a combination of nonterminal and/or terminal symbols.

  For example, the production '$p : V \rightarrow drop$' may be read as: the nonterminal symbol V (for verb) may derive or can be rewritten as the terminal symbol 'drop' (a verb), and the production '$p : NP \rightarrow Det\ N$' may be interpreted as: the nonterminal symbol NP (for noun phrase) can be rewritten as the nonterminal Det (determinant) followed by the nonterminal N (noun).
- $C \in V_{NT}$ is the start symbol to produce a sentence or command. It is a nonterminal symbol and all the sentences produced by the grammar are derived from C (note that the grammar needs an initial symbol to begin with).

The phrase-structured grammars can be augmented to represent additional information that is important to understand natural-language commands. This information can be incorporated in a grammar by assigning to each symbol (terminal or nonterminal) in the language not just a syntactic category such as verb or adjective but *attributes* that take *values*. This way, we have the so called *augmented* phrase-structured grammars. So for example, the word 'drop' could have the following two basic attributes: *category*, with the value *verb*; and *number*, with the value *singular*. To obtain the translation of a command into the executable instructions to the robot, the word may have additional attributes. For example, for the word 'drop' we can define the attribute *command* with the value *open_gripper(x,y)* where x and y are the coordinates of the point where the operator wants the robot to 'drop' an object. These coordinates depend on the values of the object attributes which appears in the command (for example in the command 'drop the box in the middle of the table', the coordinates x and y are the values of the corresponding attributes of the object 'the middle of the table').

These attribute-value pairs can be represented in matrix form in which the first column contains the attributes and the second one contains the corresponding values. This way, during the parse of a sentence these matrix structures are combined or *unified* using syntactic and semantic analysis algorithms obtaining the command that the robot has to execute.

For more details in grammars, parsing and semantic analysis the reader is referred to texts that deal specifically with these topics (for example, [19, 20]).

7.2.3 The Natural Language Tool

The basic objective of the designed natural language interface is to allow an untrained human operator to teleoperate a robotic arm through a semantics that reflects the complex kinematics and dynamics involved in the tasks performed by the robot.

In order to program a robotic arm using natural language, an intermediate language should be defined, including all the actions that the robot can execute. This intermediate language is equivalent to the target languages in database applications. In fact, this intermediate language is the language to which the system will translate the input command, so it can be considered the "target" language for the natural language subsystem. In our architecture, the instructions have been divided in three categories, depending on the type of action to execute. These categories are: (1) Movement instructions; (2) Database access instructions; and (3) Control structures. In the design of the intermediate language, the main objective is to achieve generality. We can consider the intermediate language as a model for the teleoperated robot using natural language.

Once the system has obtained the transcription of the voice command in an ASCII text, the natural language understanding module interprets the sentence and translates it into the intermediate control language. The method consists of programming a grammar or specific transition network [19] for the application and to perform the four analysis phases outlined in section 7.2.2:

- Lexical analysis: the identification of the minimum units from the input command;
- Syntactic analysis: the identification of the syntactic structure of the sentence, obtaining a syntactic tree; and
- Semantic and pragmatic analysis: to obtain the interpretation and final translation of the command.

7.3 Real-Time Control: Error Feedback Using Natural Language

This section presents methods for error feedback processing using natural language. Two different methods have been used such that it is possible to process the feedback from errors produced by a teleoperated robot, using a voice interface with natural language processing capabilities. The solutions proposed in this chapter are motivated by the following questions: (1) How can the feedback be processed during the execution of a teleoperated command, expressed using a voice interface, and (2) how can the system use the information provided by the feedback process so that the robot behaves in the way desired by the operator in successive executions of the task.

We do not consider the automatic recovery from errors [21]. The feedback subsystem using natural language described in this section has two main effects:

(i) The correction in real time of the robot's current action; and
(ii) learning of derived information obtained from the correction such that this information can be used in later executions.

In this section it will be assumed that the system has already a set of tasks that the robot has learned how to execute, or that in some way it has a list of sequential actions associated to each one of the tasks of this set. It is also assumed that most of these tasks depend on a set of parameters, that is, their execution depends on an n-dimensional vector $\mathbf{p}$ of m parameters, $\mathbf{p} = \{p_1, p_2, \ldots, p_m\}$. These parameters can be positions at each step of the execution of the task, speeds, forces, etc.

The value of these parameters is what will be adapted or learned through the feedback. The tool considered in our system can process two types of feedback:

(i) ***Position feedback***, which corresponds to commands to the robot in cases in which some position (final or partial) has not been completely satisfactory. Typical commands that fall in this category are: "*... more to the left ...*", "*... much more to the left ...*", "*... a little more to the right ...*", etc.
(ii) ***Accuracy feedback***, which corresponds with commands that refer to the value of some magnitude (forces, speed, etc.) and they can be expressed using natural language commands such as "*... press with more care ...*", "*... not so slowly ...*", etc.

The automatic adaptation algorithm consists of the following steps:

(i) The operator speaks a command, and then the robot executes this command using m functions f_i $(i = 1, 2, \ldots, m)$ to determine the values of the m parameters $(p_1, p_2, \ldots, p_m)$.
(ii) Depending on the execution, the operator speaks feedback commands.
(iii) The robot processes the feedback commands in two steps:
    a) The robot executes an immediate action according to the feedback command in order to correct in real time the position or accuracy parameter.
    b) The system *adapts* the internal representation of the parameter for future executions of the command.

Depending on the type of functions f_i used in this algorithm, two similar techniques can be distinguished: fuzzy representation techniques and stochastic representation techniques. The main differences between these two techniques reside in the knowledge representation (functions f_i) and the specific algorithm to update this representation, but the main idea is very similar. Secs. 7.3.1 and 7.3.2 describe these two techniques.

7.3.1 Fuzzy Representation Techniques

In the previous algorithm, before feedback takes place on the robot's behavior, the operator should speak a command. When the robot has to decide the value of the parameter vector $\mathbf{p}$ associated with this command, a sampling algorithm is

used through the function f_i. An interval $[x_{\min}, x_{\max}]$ is defined for the generation of the values.

Once the robot has chosen the value of the parameter from the current function that represents the knowledge, the user can speak correcting commands associated to that parameter. As reply to a correcting command, the robot executes an immediate action and it adapts the internal representation of the parameter depending on the command. Therefore, the nature of the feedback command determines the adaptation algorithm of the functions f_i. We consider two types of feedback commands: position and accuracy.

Position Feedback

The position feedback in natural language should have an immediate effect on the robot's position, besides modifying the function associated to the position parameters for that task. For example, after a feedback like "... *a little more to the left* ...", the robot will move a longitude toward the left and it will also adjust the coordinates for the next time it executes the task.

The magnitude of the displacement as a consequence of the position feedback depends on the specific feedback parameters that the operator has spoken. A simple way of modelling these displacements is to define n different constants $c_1, c_2, \ldots, c_n$ for each position feedback category, such that:

$$c_1 < c_2 < \ldots < c_n$$

The commands that generate relatively "very small" movements will come defined with the constant c_1, those that generate relatively "small" movements with the constant c_2, those that generate "middle" movements with the constant c_3, "big" movements with the constant c_4, "very big" movements with the constant c_5, and so on. The decision about the value that should have the variable n and the concrete values that should have the n constants c_i depends on the nature of the parameter and on the specific task and they are determined by means of a specific design or in the learning phase of the tasks.

In the fuzzy representation technique, the values of each parameter p_i in the command are represented by a fuzzy membership function in such a way that the specific value of the robot's displacement is multiplied by its dispersion. These membership functions are centered around a real number, with arbitrary functions to both sides of the center, and they are called *Left-Right fuzzy numbers* $(L-R)$, whose membership functions are defined as:

$$f_i(x) = f(x; a, \alpha_L, \alpha_R)_{LR} = \begin{cases} L\left(\frac{a-x}{\alpha_L}\right), & \text{if } x \leq a \\ R\left(\frac{x-a}{\alpha_R}\right), & \text{if } x \geq a \end{cases} \tag{7.1}$$

where a is the center of the fuzzy set, α_L and α_R are positive real numbers which represent the dispersion of the function, and R and L are two functions that satisfy the following conditions:

(i) $R(0) = L(0) = 1$, and
(ii) R and L are non-increasing in the interval $[0, \infty[$.

To decide the change in the robot's position as a consequence of the feedback with voice, as it has been indicated before, the constant c_i is multiplied by the dispersion corresponding to the membership function, given directly by the constants α_L and α_R in the case of $L-R$ fuzzy numbers in (7.1). If triangular or trapezoidal membership functions are used, then the dispersion is given by the distance between the center and the extremes of the triangle or trapezium.

Besides causing a change in the robot's position, this feedback type should cause a change in the generation of the parameter for future executions of the same command expressed in natural language (step 3b in the previous algorithm). In order to achieve this objective, the membership functions should be modified in the following way:

- The center a of the membership function f_i is updated as the last value of the corresponding parameter p_i after the feedback.
- The variation of the dispersion (given by the parameters α_L and α_R) is computed as follows:

$$\alpha_R(i+1) = \sqrt{\alpha_R(i)} \tag{7.2}$$

$$\alpha_L(i+1) = \sqrt{\alpha_L(i)} \tag{7.3}$$

  This correction is made assuming that after several executions and corrections by the user, the dispersion of the function should tend to decrease, since the more corrections carried out, the bigger "reinforcement" of the learning of the parameter. A way of getting this is making the new α_L and α_R values equal to the square root of the previous values. In this way it is possible to decrease the dispersion in the selection of the parameter whose value has been corrected.

Accuracy Feedback

As it has already been mentioned, an accuracy feedback is the speaking of interaction commands in natural language that refer to the value of some magnitude (forces, speed, etc.). These commands are expressed with commands in natural language such as "*... press with more care ...* ", "*... not so slowly ...*", etc. This feedback doesn't cause an immediate change in the position and the robot's current state, but rather it causes a change in the function that is used to generate the magnitude to which refers the command. As in the previous case, the change can be reflected changing the dispersion of the membership function of the fuzzy number directly. If the command refers to the need of increasing the value of a magnitude, then the dispersion should be increased (for expressions like "*... more quickly ...*").

On the other hand, if the command refers to the need of decreasing the value of a magnitude, then the system should decrease the dispersion explicitly (for example, for expressions like "*... with less force ...*". The change that is made to the dispersion should be proportional to the current values of the dispersion,

so that the change takes place in a controlled way. A form of getting this is to increase or to decrease the dispersion according to the absolute value of the first derivative of the dispersion with respect to one of the parameters of the membership function (for example, with respect to the center), so that an increase or decrease following the direction of the gradient takes place.

7.3.2 Stochastic Representation Techniques

Another considered possibility to represent the imprecise information in the feedback to the robot using the speech interface is the direct use of probability density functions. For one-dimensional tasks, simple probability distributions of one variable can be used, $f(x)$, where x is the value of the parameter, and f is the probability density function. Using these ideas, a stochastic approach can be designed as an alternative representation of the parameters that may be adjusted. The robot will use as working parameter values of p that belong to an interval around the mean of the distribution, with a dispersion that is given by the variance.

The choice of the function f depends on the robot, the environment and the specific task. In the work described in this paper a well-known function has been used, the beta function or distribution whose probability density is given by [11]:

$$f(x) = \frac{g(\alpha+\beta)}{g(\alpha)g(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad \alpha > 0,\ \beta > 0,\ 0 < x < 1 \tag{7.4}$$

where g is the gamma distribution, which is defined as $g(x) = \int_0^\infty e^{-t}t^{\alpha-1}dt$.

This function is used in the work presented in this paper to represent probabilistically behaviours of the robot that depend on one or several parameters. Depending on the values of α and β, the function will represent different behaviours. If $\alpha = \beta = 1$, then all the values in the interval $(0, 1)$ are equally probable, representing the idea that the operator doesn't care too much the value of the parameter while it belongs to a specific interval. In the case of position feedback described in the previous sections, this is translated to the fact that it is not relevant the exact point in which the position of a tool is placed, in the range of its longitude (that is, in the range of the function f). For the values $\alpha = \beta = 1$ a uniform distribution in the interval $(0, 1)$ is obtained, that is usually the initial distribution for the task.

7.4 Experimental Results

The work environment of the robotic arm that will be considered in this section consists of a table, a set of pieces and a shelf in which the pieces can be placed (see Fig. 7.1). The operator should be able to communicate with the robot to perform simple assembly tasks. These assembly tasks include simple subtasks such as moving the pieces, sorting, or storing them, etc. We assumed that the robot has already learned how to perform these tasks and the corresponding procedures are stored in a knowledge database.

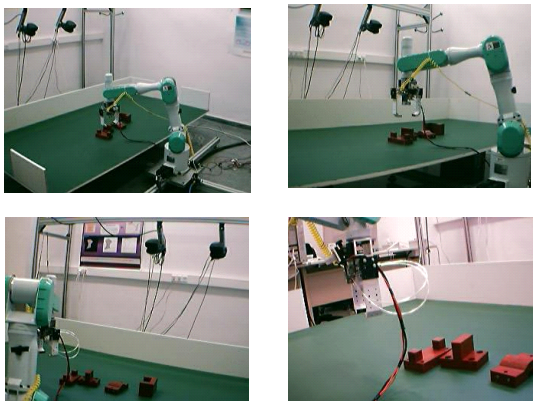

Fig. 7.1. Different images corresponding to the robotic arm working in the remote environment using the natural language interface

The group of techniques described in previous sections has been evaluated with a teleoperated robot as the one described. In Table 7.1 a sequence of values of the parameter **p** is shown. This parameter represents the distance to the origin of coordinates considered in the environment for a command of placement of a piece with position feedback, using possibilistic techniques with $L - R$ fuzzy numbers.

In Table 7.2 results from the same experiment are shown, using the beta probability distribution.

Table 7.1. Position feedback with possibilistic techniques, using $L - R$ fuzzy numbers

p	Position feedback
0.500	*more to the right*
0.725	*much more to the right*
0.911	*a little more to the left*
0.886	*a little more*
0.780	*OK*

Table 7.2. Position feedback with stochastic techniques, using the beta distribution

p	Position feedback
0.500	*more to the right*
0.300	*a little more to the right*
0.401	*a little more to the left*
0.381	*a little more*
0.751	*OK*

In Fig. 7.2 an example the sequence of membership functions is shown, obtained as a consequence of the feedback to the robot using the vector of feedback commands (R_1, R_2, R_8, R_4), obtained from the following list of commands:

- R_1: *more to the right*
- R_2: *much more to the right*
- R_3: *a little more to the right*
- R_4: *a little more*
- R_5: *a little less*
- R_6: *more to the left*
- R_7: *much more to the left*
- R_8: *a little more to the left*

In the experiments five constants c_i have been used for the decision of the robot's displacement from its current position, with the following values: $c_1 = 0.02$, $c_2 = 0.08$, $c_3 = 0.11$, $c_4 = 0.19$, $c_5 = 0.25$. These values are quantified depending on the range of the parameter that is the object of the feedback. In the case of the parameter whose membership functions are shown in Figure 7.2, the values taken by the parameter are in the interval $[0, 1]$. The membership functions can be modeled with the use of fuzzy numbers using identical definitions for the functions L and R, and they are given by the lineal function $L(x) = R(x) = -x+1$.

This function has the two conditions of the definition given previously for the fuzzy numbers ($R(0) = L(0) = 1$) and both are non increasing. All the experiments start with the value for the mean $a = 0.5$. The initial values of the dispersions have been $\alpha_L = \alpha_R = 2$. The initial value of the parameter for the execution of the command is $x = 0.5$ and the initial functions for the representation of the membership functions corresponding to the fuzzy number associated to the parameter are the following: $L\left(\frac{0.5-x}{2}\right) = 0.5x + 0.75$, and $R\left(\frac{x-0.5}{2}\right) = -0.5x + 1.25$.

Fig. 7.2 shows a sequence of probability distribution functions obtained after the processing of feedback position commands, indicating successive displacements toward the right. It can be observed that, besides moving the mean of the distribution, the process also decreases the standard deviation to reinforce the fact that feedback has taken place on the associated parameter, reducing in consequence the effective range in which the sampling algorithm obtains the values. The default initial parameters for the distribution beta are always $\alpha = \beta = 10$ (except for the cases in which explicitly a mean different from 0.5 is indicated). Later on, as a consequence of each feedback command, new values are calculated in such a way that the mean of the distribution approaches to the new obtained value, and such that the standard deviation diminishes approximately according to the constant $1/k$, where $k = 2$ in the lines of Figure 7.2. Note that since the values of α and β are integers, the solution to the equation $\sigma_f(i+1) = (1/k)\sigma_f(i)$ can only be obtained as an integer approximation to obtain the new standard deviation and the new mean values after each feedback iteration.

The two techniques described in this paper allow the processing of the feedback expressed in natural language following the objectives that have been shown

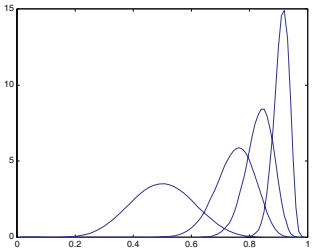

Fig. 7.2. Successive displacements toward the right of the mean of the distribution and decreasing of the standard deviation as a consequence of feedback commands

in this section. However, the fuzzy logic technique presents the advantage of being more intuitive in the definition and interpretation of the obtained representations of the parameters, besides being the most appropriate for a real time system, since the required computational complexity is smaller. The stochastic technique presents the advantage of allowing a more precise feedback of the parameters. Only in the cases in which the task carried out by the operator requires more numeric precision is justified the use of the stochastic technique, keeping in mind the biggest required computational cost.

7.5 Conclusions

The main conclusion of this section is that both representation methods allow the implementation of a feedback system that is very natural to a human operator, specially indicated when programming a robot to perform high-level tasks. It has been also shown that the feedback and correction using natural language commands is adequate for the application to a real-time teleoperated system.

The results obtained with the design of the nucleus of the system that has been shown in this section show that the natural language processing method is the most appropriate for the natural communication with a robot in real time. In particular, the definition of the intermediate language facilitates the design of the syntactic and semantic rules that define the natural language, as well as the augmented transition networks. The lexicalist paradigm used in the design of the lexicon has been the most efficient method for processing in real time. The augmented transition networks are suitable for the initial design of the interface, and for the design of the simple linguistic structures.

The voice assistance tool presented allows the user to interact with the robot in a natural way (using natural language or voice commands), and allows also the adaptation of the system from real-time feedback in natural language. The results obtained with this tool shows that the natural language processing method is the most appropriate for the natural communication with a robot in real time. In particular, the definition of the intermediate language facilitates the design of the syntactic and semantic rules that define the natural language.

References

1. T.B. Sheridan. *Telerobotics, Automation and Human Supervisory Control*. MIT Press, Cambridge, Massachusetts, 1992.
2. M. Decreton. *Teleoperation: Numerical Simulation and Experimental Validation*, chapter Nuclear Teleoperation. Particular Challenges in Decommissioning Applications. Kluwer Academic Publishers, Boston, 1992.
3. J.M. Sabater, J.M. Azorin, O. Reinoso, R. Neco, and N. Garcia. Dynamic virtual environment to test teleoperated systems with time delay communications. *Journal of Robotic Systems*, 22(4):167–181, 2005.
4. R. Neco, O. Reinoso, N. Garcia, and R. Aracil. A structure for natural language programming in teleoperation. In *Sixth International Conference on Control, Automation, Robotics and Vision*, 2000.
5. C. Crangle. Conversational interfaces to robots. *Robotica*, 15(1):117–127, 1997.
6. M.K. Brown, B.M. Buntschuh, and J.G. Wilpon. Sam: A perceptive spoken language understanding robot. *IEEE Trans. on Systems, Man and Cybernetics*, 22(6):1390–1402, 1992.
7. Y. Anzai. Human-robot-computer interaction: a new paradigm of research in robotics. *Int. J. Adv. Robotics*, 8(4):334–358, 1994.
8. R. Prasad, H. Saruwatari, and K. Shikano. Robots that can hear, understand and talk. *Advanced Robotics*, 18(5):533–564, 2004.
9. J. Biing-Hwang and F. Sadaoki. Automatic recognition and understanding of spoken language - a first step toward natural human-machine communication. In IEEE, editor, *Proc. of the IEEE*, volume 88, pages 1142–1165, 2000.
10. M.S. Eid and C. Moghrabi. Natural language interfaces as expert systems for industrial applications. *Computers in Industrial Engineering*, 31(3/4):843–847, 1996.
11. C. Crangle and P. Suppes. *Language and learning for robots*. Center for the Study of Language and Information (CSLI), Stanford, 1994.
12. S. Carberry, J. Chu-Carroll, and L. Lambert. Modeling intention: Issues for spoken language dialogue systems. In *Proc. of the International Symposium on Spoken Dialogue*, pages 13–24, 1996.
13. M. Ferre, J. Macias-Guarasa, R. Aracil, and A. Barrientos. Voice command generation for teleoperated robot systems. In *7th IEEE International Workshop on Robot and Human Communication (RO-MAN'98)*, volume 2, pages 679–685, 1998.
14. N. Iwahashi. Language acquisition through a human robot interface by combining speech, visual, and behavioral information. *Information Sciences*, 156:109–121, 2003.
15. M.C. Torrance. *Natural communication with robots*. MIT, Cambrigde, 1999.
16. E. Alessandri, A. Gasparetto, R. Valencia, and R. Martinez. An application of artificial intelligence to medical robotics. *Journal of Intelligent and Robotic Systems*, 41(4):225–243, 2005.
17. D. Sofge, Bugajska, J.G. Trafton, and D. Perzanowski. Collaborating with humanoid robots in space. *International Journal of Humanoid Robotics*, 2(2):181–201, 2005.
18. K.R. Thorission. A mind model for multi-modal communicative creatures and humanoids. *Int. J. Appl. Artif. Intell.*, 13(4-5):449–486, 1999.
19. J. Allen. *Natural Language Understanding*. Benjaming Cummings Series in Computer Science, 1995.
20. D. Jurafsky and J.H. Martin. *Speech and Language Processing*. Prentice Hall, 2000.
21. H. Aydin, R Melhem, and D. Mosse. Incorporating error recovery into the imprecise computation model. In *Proceedings of the 7th International Conference on Real-Time Computing Systems and Applications*, 1999.