A Methodology for Creating an Adapted Command Language for Driving an Intelligent Wheelchair

Brígida Mónica Faria[1,2,3], Luís Paulo Reis[2,4], and Nuno Lau[3,5]

[1]Escola Superior Tecnologia de Saúde do Porto / Instituto Politécnico do Porto (ESTSP/IPP), [2]Laboratório de Inteligência Artificial e Ciência de Computadores (LIACC), [3]Inst. Eng. Electrónica e Telemática de Aveiro (IEETA), [4]Dep. Sistemas de Informação, Escola de Engenharia da Universidade do Minho (DSI/EEUM), [5]Dep. Electrónica, Telecomunicações e Informática da Universidade de Aveiro (DETI/UA)

Emails: btf@estsp.ipp.pt, lpreis@dsi.uminho.pt, and nunolau@ua.pt

Abstract. Intelligent wheelchairs (IW) are technologies that can increase the autonomy and independence of elderly people and patients suffering from some kind of disability. Nowadays the intelligent wheelchairs and the human-machine studies are very active research areas. This paper presents a methodology and a Data Analysis System (DAS) that provides an adapted command language to an user of the IW. This command language is a set of input sequences that can be created using inputs from an input device or a combination of the inputs available in a multimodal interface. The results show that there are statistical evidences to affirm that the mean of the evaluation of the DAS generated command language is higher than the mean of the evaluation of the command language recommended by the health specialist (p value = 0.002) with a sample of 11 cerebral palsy users. This work demonstrates that it is possible to adapt an intelligent wheelchair interface to the user even when the users present heterogeneous and severe physical constraints.

Key Words: Intelligent Wheelchair; Command Language; User Modeling; Data Analysis System.

1. Introduction

The analysis of the tasks for a human to perform, the information and technological requirements, the machine ergonomics and design are among the most interesting topics of study in the field of Human-Machine interaction.

Systems that make the bridge between users and the processes to be controlled are another key point in this area. The challenges are even greater when studying the adaptation of technology used by individuals with disabilities in order to perform tasks that might otherwise be difficult or even impossible for them.

Scientific research allowed the evolution and development of many technologies that are nowadays used in everyday life. In particular, innovations in the field of assistive technologies enabled increased autonomy and independence for human beings that, for some reason, have some kind of disability. Intelligent wheelchairs are an obvious application of the scientific work developed in the last decades on this area [1]. Moreover, these assistive technologies still are object of research and the interaction between them and the user remains an open research problem. The interaction between the Human and the IW is an important component to take into consideration.

The methods implemented in this work allowed answering several questions on the adaptation of an intelligent wheelchair that can be commanded via a multimodal interface. Another issue is related with the creation of users' profiles in order to automatically adjust the best way of driving the intelligent wheelchair. Users' classification demands data of distinct sources such as voice, physical movements like head or facial expressions or data taken from using the usual joystick that is common in electric wheelchairs. However, gathering and analyzing this type of data is still an open research problem. In order to face this problem, new multimodal data gathering and analysis methodologies were developed enabling to build a complete data gathering and data analysis system to generated an adapted command language for driving an intelligent wheelchair.

The paper is organized as follows: Section 2 briefly describes our project, the system architecture and the context of the multimodal data gathering system. Section 3 presents the Data Analysis System (DAS) and the purposed solution to give an adapted command language allowing an user to drive an intelligent wheelchair. The used algorithms are also presented. The experiments and results compose Section 4 and in Section 5 the conclusions and future work are presented.

2. Intellwheels Project

The main objective of the IntellWheels Project is to develop an intelligent wheelchair platform that may be easily adapted to any commercial wheelchair and aid any person with special mobility needs [2] [3]. Several different modules have been developed in order to allow different ways of carrying inputs (I_i) for driving the IW. These include several input devices, such as, joystick control with USB,

microphone for voice inputs, wiimote control for head movements, and a brain computer interface for facial expressions and thoughts recognition [4].

Within this work new ways of interaction between the wheelchair and the user have been integrated, creating a system of multiple entries based on a multimodal interface.

2.1 System Architecture

The IntellWheels system architecture that also enabled to conduct the experiments of user profiling and the DAS is presented in Figure 1. The system is composed by eight main modules and enables a therapist to have full control of all the IW adaptation process.

The core of the system is the new IntellWheels multimodal interface that enables the patient to fully control real and simulated Intelligent Wheelchairs, using multimodal inputs, including pre-defined input sequences that may be freely associated with any of the available outputs (wheelchair actions). The input devices may be freely connected to this multimodal interface. The multimodal interface is connected to a control module that is able to receive high-level or medium level commands from the multimodal interface and control a real or simulated wheelchair making it perform the actions corresponding to those commands (such as "go front", "turn right", "follow right wall", "stop", among others).

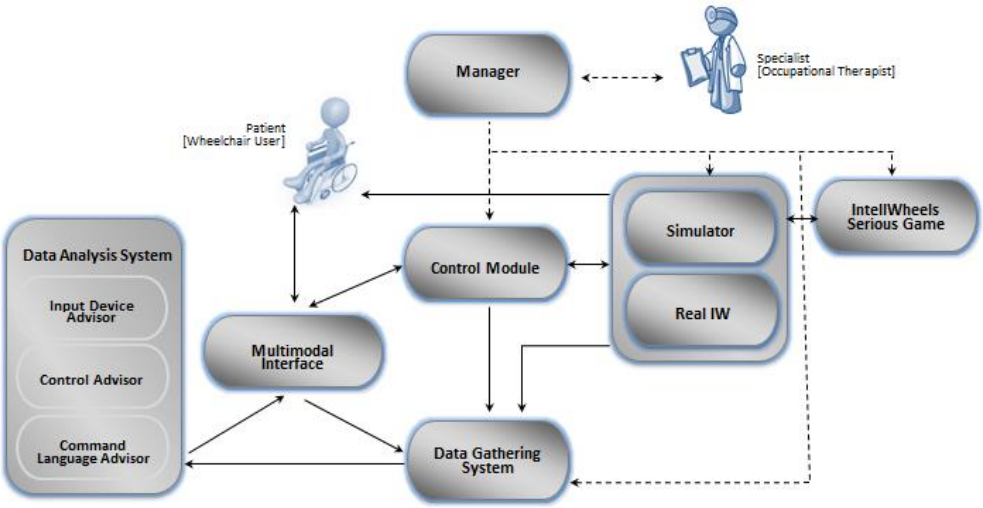

Figure 1. IntellWheels system architecture.

In order to be able to develop and conduct meaningful experiments, a serious game for intelligent wheelchair teaching and testing was built. The game permits

the definition of circuits and the placement of markers that must be collected by the user in order to gain points. It also enables gathering other performance measures such as the time and the precision of the trajectory of the users performing the circuit [4].

In order to be able to extract user profiles and adapt the user interface to the users' profiles several other applications were developed. One of these applications consists on a complete data gathering system that is able to gather the data available on: the multimodal interface; control module; simulated wheelchair; real wheelchair and serious game, and then synchronize all this data and freely select the values to record in appropriate files in order to be further analyzed by the data analysis applications. A user profiling application was also created in this context in order to be able to conduct controlled experiments with each user in order to analyze their capabilities of performing each type of possible input in each of the available input devices [5][6][7].

Based on the user profiling and associated data gathering system, a DAS was developed enabling the analysis of users' capabilities when performing each type of input and when driving the IW with different input combinations. Beyond data analysis, this module is able to advise, in a simple manner, the best control mode for each user and to specify a command language adequate for each user.

A manager module was also developed in order to be able to perform a large set of experiments using the developed user profile extraction methodology and the set of implemented applications. This manager allows to launch all the applications, perform user profiling tests, define the scenario to be used, the circuit to be performed, the control modes to be tested and the data to be gathered and analyzed.

2.2 Multimodal Data Gathering and User Profiling

In order to be able to extract patient models and also environment models, a complete multimodal data gathering system was implemented. Based on the IntellWheels prototype and using the real and simulated environments, the work was focused on planning appropriate data gathering and DAS that enable the creation of an adapted interface and command language adjusted to the patient and where information about the environment is also considered.

A user profiling module was designed and implemented. This module, along with the IntellWheels DAS, helps in the process of giving the more adequate input device for driving the wheelchair. Initially a set of tasks and actions was defined to be executed by the user. A wizard, or more specifically the profiling component of the multimodal interface (Figure 2), includes simple tasks that can be performed with input devices and that permit an evaluation of the user ability to use that device [8].

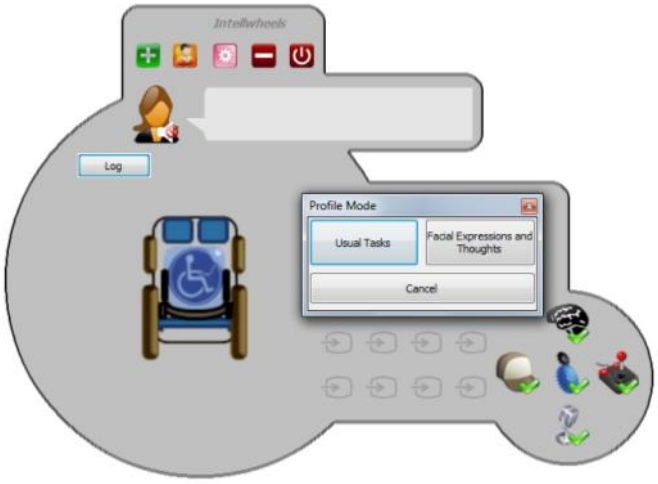

Figure 2. Starting user's profile module.

The performance of each task was collected and the specialists (occupational therapists) were integrated in the process to confirm the correct classification. The data analysis system advises the user about the best suited input(s) device(s) and command language. The system also records the information about each user and if the user wants to update the information.

3. Data Analysis System Implementation

The IntellWheels DAS is the component that advises the user on the best input device for driving the intelligent wheelchair. Moreover, the DAS, using information from the pre-processing module of multimodal data fusion, is capable of extracting the most relevant information from the patient data gathering system application (profile module) which enables fast generation and configuration of the interfaces. The system advises the best options for driving the intelligent wheelchair including the best set of input sequences and their association with the available commands. The best choice for the command language is going to consider the best recognition combination, the best efficiency and the best

intuitiveness combination. The objective is to have the best association of inputs and commands, considering the user characteristics, to drive the intelligent wheelchair. This means that it is necessary to first define a set of commands. For example using five commands, as in Table 1, associated to an input sequence set:

Table 1. Input sequences associated with commands

Inputs sequences	Commands
Press button 1	"Go Forward"
Press button 1 – Press button 3	"Go Back"
Press button 1 – Tilt the head to the right side	"Turning Right"
Say "Go" – Say "Left"	"Turning Left"
Smile	"Stop"

Next the DAS requirements, in order to provide the best interface for a specific user, are going to be presented.

3.1 Requirements

The IntellWheels Data Analysis System has several requirements that should be fulfilled:

- Enable multiple input devices – the command language should be able to include inputs from different input devices so that it has a higher range of facilities for driving the IW;
- Maximize user performance in driving the IW – the objective is to present a solution where the performance, usability and safety is maximized;
- Be adapted to multiple users with distinct disabilities – the IW should be available and adapted to different users and to different disabilities;
- Fast response to user commands – the time between starting an input sequence and executing the corresponding command should be minimized;
- Associate several distinct input sequences with similar performance to the same command – the fulfilment of this option allows a user which degrades for example one of his abilities, to be able to drive the IW with another set of options;
- Intuitiveness between the associations of the input sequences and the commands – the user should use input sequences that are user friendly, for example saying "Forward" should mean that the wheelchair should go

forward or “Blink the right eye” should mean that the wheelchair should turn right instead of going left.

In order to explain the proposed solutions the definition and formalizations of confusion matrix for each input device are presented in the next subsection.

3.1 Inputs’ Confusion Matrix and Measures

The data acquisition system in the Profile Module also provides the information about what was asked and what was recognized by the system. For that reason it is possible to obtain a confusion matrix for each input and for each input device.

The confusion matrix of each input device can be designated as in Equation 1:

$$CM_{ID} = (n_{ij})_{\substack{i=1,\ldots,N \\ j=1,\ldots,N}} \quad (1)$$

where i designates the lines, j the columns of the matrix, n_{ij} is the number of times that I_j is recognized as I_i and ID is the input device.

For example Table 2 represents the confusion matrix with the inputs that can be expressed saying “Go”, “Left”, “Right”, “Back” and “Stop”.

Table 2. Confusion matrix defined for the microphone

$CM_{Microphone}$		True I_1 (“Go”)	I_2 (“Left”)	I_3 (“Right”)	I_4 (“Back”)	I_5 (“Stop”)
Predicted	I_1 (“Go”)	n_{11}	n_{12}	n_{13}	n_{14}	n_{15}
	I_2 (“Left”)	n_{21}	n_{22}	n_{23}	n_{24}	n_{25}
	I_3 (“Right”)	n_{31}	n_{32}	n_{33}	n_{34}	n_{35}
	I_4 (“Back”)	n_{41}	n_{42}	n_{43}	n_{44}	n_{45}
	I_5 (“Stop”)	n_{51}	n_{52}	n_{53}	n_{54}	n_{55}

For each input device confusion matrix it is possible to calculate the recall and precision of each input. The recall of each input is defined as the probability of a true input being correctly classified and can be calculated as in Equation 2:

$$rec_i = \frac{n_{ii}}{\sum_{m=1}^{N} n_{mi}} \quad (2)$$

where n_{mi} is the number of times that I_i is recognized as I_m and N the number of inputs. The precision of each input is defined as the probability of a predicted input represents that true input and can be calculated as in Equation 3:

$$prec_i = \frac{n_{ii}}{\sum_{m=1}^{N} n_{im}} \quad (3)$$

where n_{im} is the number of times that I_m is recognized as I_i and N the number of in-puts. It is important to refer that in the concrete problem of giving an adapted command language, an extra case representing when other distinct input was predicted, was added in the predicted categories.

It is possible to combine the recall and precision of each input using, for example, the arithmetic mean or a more adequate measure that uses the harmonic mean called the F-measure [9]. This measure gives high values only when both precision and recall have high values. Equation 4 presents the general definition of the $F_{\beta i}$-measure of each input:

$$F_{\beta i} = \frac{(\beta^2 + 1) \times prec_i \times rec_i}{\beta^2 prec_i + rec_i} \quad (4)$$

where $0 \leq \beta < +\infty$ is a parameter that controls the balance between the recall and the precision. In the experiments recall and precision are evenly weighted therefore it was used the value 1 for β.

3.2 Command Language

In order to generate a command language adapted to a given user several points should be taken into account: the time efficiency, the recognition probability of an input sequence and the intuitiveness of an input sequence to be associated to a command. Figure 3 shows the quantifiable criteria used for the command language definition.

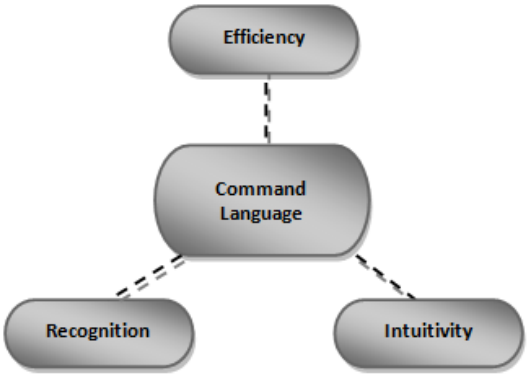

Figure 3. Criteria used for the command language.

Next, these three points are going to be presented in more detail using the formalization of the measurable criteria.

3.2.1. Time and Time Efficiency

Assuming that a sequence of inputs S_i can be formalized as $I^{(i,1)}\ I^{(i,2)}\ I^{(i,3)}...I^{(i,Ni)}$, where each $I^{(i,N_i)} \in \{I_1, I_2, ..., I_k\}$ and a single command can be associated to a final sequence that produces an action, the time to generate a command is composed by a component of time to select the inputs and by the time taken by the command to generate a visible action or time of output ($t_{timeout(i)}$). The total time for a particular command to be used has the Equation 5:

$$t_{S_i} = \sum_{k=1}^{N_i} t^{ID}_{I^{(i,k)}} + t_{timeout(i)} \tag{5}$$

where k is the number of each of the inputs used in the sequence, S_i the identification of the sequence i and N_i the total number of the inputs of sequence i. Therefore it is possible to determine the total time for all the commands necessary to drive the intelligent wheelchair as in Equation 6:

$$T_C = \sum_{j=1}^{C_j} t_{S_j} \tag{6}$$

where C_j is the number of commands in the command language.

The time efficiency can be defined as a function of time, if more time is necessary for a command to be used then that command is less efficient. It is possible to formalize this function as in Equation 7:

$$\begin{aligned} eff : [0,+\infty[&\rightarrow [0,1] \\ t_{S_i} &\mapsto \frac{1}{t_{S_i}+1} \end{aligned} \tag{7}$$

The total time efficiency (Equation 8) is the sum of all the efficiency values of the commands that compose a command language:

$$T_{C_{eff}} = \sum_{j=1}^{C_j} eff\,(t_{S_j}) \tag{8}$$

3.2.2. Sequence Recognition

It is also possible to define and calculate the sequence S_i recognition value. Assuming the independence of recognition of the different inputs in a sequence,

the sequence S_i recognition value is the product of the F-measure values as in Equation 9.

$$regS_i = \prod_{k=1}^{N_i} F_{I^{(i,k)}}^{ID} \tag{9}$$

where $F_{I^{(i,k)}}^{ID}$ is the F-measure value in the position of the principal diagonal of the input $I^{(i,k)}$ be in use in the sequence and for a specific input device (*ID*). The total recognition value of a set of commands can be determined by Equation 10:

$$T_{reg} = \sum_{j=1}^{C_j} regS_j \tag{10}$$

where C_j is the number of commands in the command language.

3.2.3. Intuitiveness

Another concept that should be analyzed is the intuitiveness of a sequence of inputs. In order to have values similar to the efficiency and recognition, it was defined that an input sequence, associated to a given action, can have a value of intuitiveness between 0 and 1. The value of 1 means that the input is very typical for performing that command and a value of 0 means that the input typically is associated with an opposite command. For example, if a sequence is composed of a single input such as saying "front" and the action of the wheelchair associated is go forward then the intuitiveness value may be 1. If the same input is associated with the command that makes the IW going back then the intuitiveness will be 0. Table 3 presents an example of the intuitiveness of several voice inputs.

Table 3. Intuitiveness for the voice inputs

		I_1	I_2	I_3	I_4	I_5	I_6	I_7
		Go	Left	Right	Back	Stop	Front	Forward
Commands	Forward	1	0	0	0	0	1	1
	Left	0	1	0	0	0	0	0
	Right	0	0	1	0	0	0	0
	Back	0	0	0	1	0	0	0
	Stop	0	0	0	0	1	0	0

The intuitiveness of a sequence composed by two or more inputs can also be obtained by the product of the intuitiveness of each input. In fact, an example is a sequence composed of two inputs such as say "front" "front" then the intuitiveness in this case it is also 1 when the objective is to drive the wheelchair

forward. In particular, the intuitively is chosen by the user and for example the users of the system (Table 3) intuitively associated “Go” with the meaning “Go forward”.

3.2.4. Command Language Implementation

In order to obtain the best performance, the command set that maximizes the sequence recognition, the intuitiveness and time efficiency should be obtained.
Basically, a command language adapted to the user should be found, that maximizes the function composed by the total time efficiency, total recognition and intuitiveness:

$$\arg\max_{T_{eff},T_{reg},T_{\text{int}}}(\alpha T_{eff} + \beta T_{reg} + \gamma T_{\text{int}}) \tag{11}$$

where α, β and γ are parameters that could be adjusted. The optimization may be performed by any type of optimization algorithm with emphasis on iterative meta-heuristics such as basic hill-climbing [10], simulated annealing [11], tabu search [12] or genetic algorithms [13]. For the implementation, in order to show the concept, a modified hill-climbing algorithm was implemented, mainly due to its simplicity. The pseudo-code and the details of the implementation are subsequently explained. First the user abilities on using several inputs are captured with the profile module and the recognition values are obtained for all the available inputs. The time taken to execute each input sequence is also captured and the efficiency of performing the inputs is calculated. The degree of intuitiveness was indicated initially by the user or by a specialist. Figure 4 details the algorithm implementation.

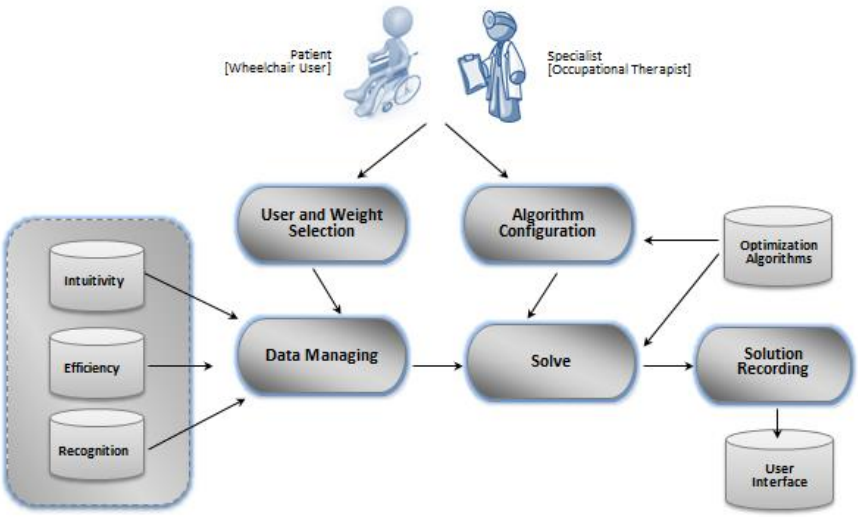

Figure 4. Command language advisor implementation.

The system starts by reading the selected user recognition and efficiency data and the intuitiveness data for the set of available inputs and commands. After selecting and configuring the optimization algorithm, the system solves the optimization problem, as previously defined, using a given meta-heuristic and subsequently recording the solution so that it can be used on the context of the multimodal interface. Hill-climbing was selected for performing the experiments on this work. However, it is easy to extend the system for using other optimization algorithms such as simulated annealing or genetic algorithms.

The pseudo-code for the optimization process is next detailed. For this implementation only voice, joystick and wiimote inputs were used. However, the system may be easily extended with further input devices using them exactly as the three included in this version.

Algorithm 1: Command_Language_Advisor(userName, NID, NM, NC, NS, algId), solution, best

```
1.  inputs:
2.     userName – User name (that enables to consult user characteristics and data)
3.     NID – Number of input devices. 3 Input devices were used (joystick, voice and wii)
4.     NM – Maximum number of inputs per input device. It includes (NV, NJ, NW)
5.          as the maximum number of Voice Inputs, Joystick Inputs and WIImote Inputs
6.     (NC, NS) – Number of available commands and maximum of Inputs in a sequence
7.     algId- Algorithm identification enabling to get all algorithm parameters
8.  outputs:
9.     solution – Solution containing one input sequence for each command
10.    best -  Best solution evaluation
11. begin
12.    id ← getID_usersFile(userName)
13.    weights = (w_rec, w_time, w_intu) ← readfile_user_weights(id)
14.    rec = (rec_voi[NV], rec_joy[NJ], rec_wii[NW] ← readfile_recognition(id)
15.    time = (time_voi[NV], time_joy[NJ], time_wii[NW]) ← readfile_efficiency(id)
16.    intu = (intu_voi[NC][NV], intu_joy[NC][NJ], intu_wii[NC][NW]) ← readfile_intuit(id)
17.    alg = (algType, param, maxIter, maxNoImp, neighbourF) ← Readfile_alg(algId)
18.    solution ← random_solution(alg, (NC, NS), (rec, time, intu))
19.    best ← evaluate_solution(solution, weights, rec, time, intu)
20.    currBest ← best
21.    currSolution ← solution
22.    it ← 0
23.    noimp ← 0;
24.    while it<maxIter ∧ noImp<maxNoImp do
25.       solNew ← neighbour_solution(neighbourF, currSolution, (NC, NS, NID))
26.       if repeated(solNew) then
27.          val ← -∞
28.       else
29.          val ← evaluate_solution(solNew, weights, rec, time, intu)
30.       endif
31.       if solution_change_criteria(alg, val, best) then
32.          currSolution ← solNew
33.          currBest ← val
34.          noImp ← 0
35.       endif
36.       if currBest > best then
37.          best ← currBest
```

```
38.          solution ← currSolution
39.       else
40.          noImp ← noImp + 1
41.       endif
42.       alg ← update_alg_parameters(alg, it, currSolution)
43.       it ← it + 1
44.    endwhile
45.    return (solution, best)
46. end
```

The algorithm receives the user name, number of input devices (three on this version: voice, joystick and wiimote inputs), number of available commands and the maximum size for the input sequences. It also receives the algorithm id that enables to consult the algorithm type and parameters. The algorithm outputs a solution that associates to each command an input sequence trying to maximize the evaluation function considered. It also outputs the evaluation value achieved for the best solution found. The solution structure is depicted in Table 4. The solution is basically a matrix of number of available commands (NC) (for example: "Front", "Left", "Right", "Back" and "Stop") and NS inputs forming the corresponding input sequence used to trigger that command. Each cell of the solution matrix may be NULL (in case the sequence used is shorter than the maximum number of inputs for a sequence NS) or composed by an input device and an input.

Table 4. Command Language Advisor Solution Structure

Number of maximum inputs in a sequence (NS=4)	Number of Commands (NC=5)				
	1	*2*	3	4	5
1	*wii* *2*	*voice* *1*	*wii* *2*	*voice* *1*	*voice* *1*
2	*NULL*	*joy* *1*	*wii* *3*	*joy* *3*	*joy* *3*
3	*NULL*	*NULL*	*wii* *1*	*NULL*	*joy* *3*
4	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*
	Ex: Front	*Ex: Left*	*Ex: Right*	*Ex: Back*	*Ex: Stop*

The Command Language Advisor, starts by reading all the input files containing the problem data. This includes consulting the user id, the weights to be used for the recognition, efficiency and intuitiveness (to be used on the evaluation of a given solution). The algorithm also reads the input files containing all the available data concerning the user. This includes the recognition and efficiency

vectors for all possible inputs (voice, joystick and wii on this implementation) and the intuitiveness matrix that relates the intuitiveness of using each of the inputs available on the three input types for performing each of the avail-able commands. Finally, the algorithm parameters are read from the algorithm database.

The solving process starts by generating an initial random solution for the problem, composed by a valid input sequence (composed by 1 to *NS* inputs) for each of the possible (*NC*) commands. It then evaluates the solution and saves the solution and evaluation as the best ones of those already tested. The main algorithm cycle is composed by *maxiter* iterations (or *maxnoimp* iterations without improvement). In each iteration, a new solution is calculated, that is neighbour (using the defined neighbouring function) from the present solution. Algorithm 2 displays the simple neighbour algorithm that was used in most of the experiments.

Algorithm 2: neighbour_solution(neighbourF, solution, (NC,NS,NID)), newSolution

```
1.  inputs:
2.     neighbourF – Neighbourhood function number (not used on this simple version)
3.     solution – Solution containing input sequence for each command
4.     NC, NS – Number of commands and maximum inputs in sequence
5.     NID–Number of Input Devices (nInputs(i) gives the number of inputs of an Input Device
6.  outputs:
7.     newSolution – Neighbour solution considering the neighbourhood function. The solution
8.              size is NCxNS. Each solution element is composed by two parts an input device
9.              (between 1 and  NID and an input between 1 and the number of inputs of that
10.             input device)
11. begin
12.    do
13.       newSolution ← solution
14.       neighbourType ← random(1, 2)
15.       if neighbourType=1 then
16.          ncom ← random(1, NC)
17.          do
18.             nseq ← random(1, NS)
19.          while (nseq ≠ 1 ∧ inputDevice(newSolution[ncom][nseq-1]) = NULL)
20.          clear← random(0, 1)
21.          if clear=1 ∧ (nseq ≠ NS ∧ inputDevice(newSolution[ncom][nseq+1]) = NULL ∨
22.                        nseq=NS) ∧ nseq ≠ 1 then
23.             inputDevice(newSolution[ncom][nseq]) ← NULL
24.             input(newSolution[ncom][nseq]) ← NULL
25.          else
26.             nInpDev ← random(1, NID)
27.             inputDevice(newSolution[ncom][nseq]) ← nInpDev
28.             input(newSolution[ncom][nseq]) ← random(1, nInputs(nInpDev))
29.          endif
30.       else
31.          ncom1 ← random(1, NC)
32.          do
33.             ncom2 ← random(1, NC)
34.          while (ncom1 = ncom2)
35.          for nseq=1 to NS do
```

```
36.             swap(inputDevice(newSolution[ncom1][nseq]),
37.                 inputDevice(newSolution [ncom2][nseq]))
38.             swap(input(newSolution[ncom1][nseq]),
39.                 input(newSolution[ncom2][nseq]))
40.         endfor
41.     endif
42.   while (newSolution = solution ∨ repeated_sequence(newSolution))
43.   return newSolution
44. end
```

Algorithm 2 considers two types of neighbours: (i) changing/adding/removing an input sequence associated to a command; (ii) exchanging the input sequences used for two distinct commands. The algorithm starts by copying the current solution to the new neighbour solution. It then decides which of the neighbour functions will be used (i) or (ii) with a probability of 50% each in the current simple implementation.

For applying neighbourhood (i) a command and an input sequence step are randomly selected until the step to change is a valid step (1 or a step without any valid steps executed after it). If the step to change is the last step and it is not step number 1 (that obviously may not be cleared), a probability of 50% is used to decide on clearing it. If the step is cleared both its input device and input are set to NULL. Otherwise a new valid value is randomly selected for the step input device and input. Neighbourhood (ii) is applied by randomly selecting two distinct commands and then swapping the commands input sequences, step by step. Finally the algorithm returns the new neighbour solution.

The solution is evaluated using the evaluation function considered and if it is better than the best solution found (given the solution change criteria used for the algorithm in use) then it will become the new current solution and the current best will be this solution evaluation. If the solution is better than the best solution already found the best solution (and its corresponding evaluation) will be changed for the best solution.

Algorithm 3: CL_Evaluator – evaluate_solution(solution, weights, rec, time, intu), evaluation

```
1.  inputs:
2.      solution – Solution containing the input device and inputs used for each input
3.          sequence for each command
4.      weights – Weights for recognition, efficiency and intuitiveness
5.      rec[3][NM] – Recognition matrix containing for each input device and input the
6.          recognition probabilities for a given user
7.      time[3][NM] – Time matrix containing for each input device and input the t
8.          times enabling to calculate efficiency information for a given user
9.      intu[NC][3][NM] – Intuitiveness matrixes relating each input from each input device
10.         to a given command for a given user
```

```
11. outputs:
12.     evaluation – Solution evaluation considering the evaluation function
13. begin
14.     (w_rec, w_time, w_intu) = weights
15.     evaluation ← 0
16.     for ncom = 1 to NC do
17.         recVal ← 1
18.         timeVal ← 0
19.         intuVal ← 1
20.         for nseq = 1 to NS do
21.             inpDev ← inputDevice(solution[ncom][nseq])
22.             inp ← input(newSolution[ncom][nseq])
23.             if inpDev = NULL then break
24.             else
25.                 recVal ← recVal * rec[inpDev][inp]
26.                 timeVal ← timeVal + time[inpDev][inp]
27.                 intuVal ← intuVal * intu[ncom][inpDev][inp]
28.             endif
29.         endfor
30.         evalComm ← w_rec* recVal + w_time*1/(timeVal+1) + w_intu*intuVal
31.         evaluation ← evaluation  +  evalComm
32.     endfor
33.     return evaluation
34. end
```

The command language evaluator algorithm (Algorithm 3) uses the pre-defined weights and the recognition, efficiency and intuitiveness user information to evaluate the current command language. It starts by initializing the evaluation to 0. Then, for each command in the solution it evaluates the input sequence used for that command given its recognition, efficiency and intuitiveness and the corresponding weights considered.

Each input sequence is evaluated until its end (and thus if a NULL value is encountered meaning the end of the input sequence its evaluation will be finished). For the recognition and intuitiveness, products of the corresponding values of the inputs on the sequence are used. For the efficiency, first the total time of the sequence is calculated and then Equation 8 is applied. On this algorithm version only commands without the need for timeout are considered and thus timeouts are not added.

The solving algorithm (Algorithm 1) final step consists on returning the solution found and its evaluation. The solution may then be used by the multimodal interface for enabling the user to drive the Intelligent Wheelchair.

4. Experiments and Results

The experiments were performed by a sample composed of 11 patients with cerebral palsy with the level IV (27.3%) and V (72.7%) of the Gross Motor Function Measure [14]. The mean of age was 27 years old with 64% males and 36% females. In terms of school level, 1 is illiterate, 1 has completed elementary school, 4 have completed middle school, 3 have completed high school and 2 have a BSc. The dominant hand was divided as: 82% for left, 18% for right hand. The frequency of use of information and communication technologies was also characterized: 7 answered rarely; 2 sometimes; 1 lot of times and 1 always. The aspects related to experience of using manual and electric wheelchair were also questioned. Table 5 shows the distribution of answers about autonomy and independency using the wheelchair and constraints presented by these individuals.

Table 5. Experience using the wheelchair, autonomy, independence and constraints of the cerebral palsy users.

Experience using the Wheelchair, Autonomy, Independence and Constraints					
Variables		**N**	**Variables**		**n**
Use manual wheelchair			Cognitive constraints		
	no	10		no	8
	yes	1		yes	3
Use electric wheelchair			Motor constraints		
	no	1		no	0
	yes	10		yes	11
Autonomy using wheelchair			Visual constraints		
	no	1		no	3
	yes	10		yes	8
Independence using wheelchair			Auditive constraints		
	no	1		no	11
	yes	10		yes	0

The voice inputs were organized in order to give several choices for the command language. The input options in this case were: “Go”, “Front”, “Forward”, “Back”, “Right”, “Left”, “Turn”, “Spin” and “Stop”. The five positions of the Joystick and Head Movements were set accordingly to the usual necessary positions for driving a wheelchair “East”, “North”, “South”, “West” and “South-west” (Figure 5).

Figure 5. Joystick and head movements positions.

An extension of the profiling was also created in order to record all the information available, such as facial expressions, thoughts and buttons pressed at the joystick. The command language evaluation given by the system solution was compared with the command language given by the occupational therapists. Table 6 shows the command language advised by the occupational therapists and by the DAS. In Table 6, all the directions given by wiimote and joystick refer to the most intuitive and natural directions. In order to compare the results obtained by the specialist and by DAS, the paired sample t test was applied to the mean of the solution evaluation after verifying the normality using the Kolmogorov-Smirnov test (p value = 0.114).

Table 6. Command Language Advisor Results

Patient		Ev	Command Language for Patients				
			Forward	**Left**	**Right**	**Back**	**Stop**
P1							
	Specialist	4.53	wiimote	joystick	joystick	joystick	joystick
	DAS	4.57	joystick	joystick	joystick	joystick	joystick
P2							
	Specialist	4.18	joystick	joystick	joystick	joystick	voice “stop”
	DAS	4.85	joystick	joystick	joystick	joystick	voice “go”
P3							
	Specialist	3.33	voice “forward”	wiimote	wiimote	joystick	voice “stop”
	DAS	4.51	wiimote	wiimote	wiimote	wiimote	voice “go”
P4							
	Specialist	4.50	voice “forward”	joystick	joystick	joystick	voice “stop”
	DAS	4.60	joystick	joystick	joystick	joystick	voice “stop”
P5							
	Specialist	4.14	voice “front”	wiimote	wiimote	joystick	voice “stop”
	DAS	4.40	wiimote	wiimote	voice “turn”	joystick	voice “stop”
P6							
	Specialist	4.13	wiimote	joystick	joystick	joystick	joystick
	DAS	4.38	wiimote	wiimote	wiimote	wiimote	wiimote
P7							
	Specialist	4.49	voice “front”	joystick	joystick	joystick	voice “stop”
	DAS	4.60	joystick	joystick	joystick	voice “back”	voice “stop”
P8							
	Specialist	3.51	wiimote	joystick	joystick	joystick	joystick
	DAS	4.20	wiimote	wiimote	wiimote	wiimote	wiimote
P9							
	Specialist	3.70	voice “forward”	wiimote	wiimote	joystick	voice “stop”
	DAS	4.75	joystick	joystick	joystick	joystick	joystick
P10							
	Specialist	4.11	voice “forward”	voice “left”	voice “right”	voice “turn”	voice “stop”

	DAS	4.80	joystick	joystick	voice “turn”	joystick	voice “go”
P11							
	Specialist	4.29	joystick	wiimote	wiimote	joystick	joystick
	DAS	4.30	wiimote	wiimote	wiimote	wiimote	wiimote

The results show that there are statistical evidences to affirm that the mean of the evaluation of the DAS is higher than the mean of the evaluation of the command language recommend by the specialist (p value = 0.002). In particular, from the total of 55 commands, from all the 11 patients, the data analysis system had exactly the same recommendation as the specialists in 44% of the commands and 53% of the advised commands by the DAS use the same input device to produce the command as the ones advised by the specialists.

5. Conclusions and Future Work

Many IW prototypes are being developed in several research projects, around the world, however the adaptation of their user interface to the patient is a neglected research topic. This research work aimed at tackling this problem developing a methodology enabling to dynamically adapt the IW user interface to the user’s characteristics. The DAS generates a command language adapted to the user. With this command language a more high level way of driving the intelligent wheelchair is possible the may even help users with the most severe cases of deficiency to be able to drive a wheelchair. It was also possible to conclude that the system results are very similar to the ones recommended by the occupational therapist. Also, the automatic generated command language had even better evaluation, combining intuitiveness, recognition and efficiency, than the command language recommended by the specialists. The data gathering process enables creating a data repository of user-wheelchair interaction that may be a used for several types of future studies. The data analysis system definition and development brings a new methodology for starting to use an intelligent wheelchair. Nowadays, this methodology is already used by a health institution for recognition of patients’ capabilities, and to test and train the users.

Acknowledgements: The authors would like to acknowledge APPC – Portuguese Association for Cerebral Palsy for all the help on the experiments. This work was partially supported by project QoLis - Quality of Life Platform, Nº2013/34034 QREN SI I&DT, (NUP, NORTE-07-0202-FEDER-034Ú34) and project Cloud Thinking (CENTRO-07-ST24-FEDER-002031), co-funded by QREN, “Mais Centro” program. The authors would like also to acknowledge to FCT –

Portuguese Science and Technology Foundation for the previous funding for the INTELLWHEELS project (RIPD/ADA/109636/2009) and previous funding for the PhD Scholarship FCT/SFRH/BD/44541/2008 and also for the funding for LIACC – Laboratório de Inteligência Artificial e de Ciência de Computadores (PEst-OE/EEI/UI0027/2014), IEETA – Instituto de Engenharia Eletrónica e Telemática de Aveiro (PEst-OE/EEI/UI0127/2014) and ESTSP/IPP – Escola Superior de Tecnologia da Saúde Porto – IPP.

References

1. Faria, B. M., Reis, L. P., Lau, N.: A Survey On Intelligent Wheelchair Prototypes And Simulators, WorldCist 2014, AISC 275, Vol. 1 Springer, Madeira, page 545-557, (2014).
2. Braga, R., Petry, M., Moreira, A. P., Reis, L. P.: Concept and design of the intellWheels platform for developing intelligent wheelchairs. Informatics in Control, Automation and Robotics, page 191–203 (2009).
3. Braga, R., Petry, M., P. Reis, Moreira, A. P.: A Modular Development Platform for Intelligent Wheelchair, J. of Rehabilitation Research and Development, 48 (9): 1061-1076, (2011).
4. Faria, B.M., Vasconcelos, S., Reis, L.P., Lau, N.: Evaluation of Distinct Input Methods of an Intelligent Wheelchair in Simulated and Real Environments: A Performance and Usability Study, Assist. Technology Journal, RESNA, Taylor and Francis, 25 (2): 88-98, June (2013).
5. Faria, B. M., Reis, L. P., Lau, N., Soares, J. C., Vasconcelos, S.: Patient Classification and Automatic Configuration of an Intelligent Wheelchair, Communications in Computer and Information Science 358, Springer-Verlag, page 268-282, (2013).
6. Faria, B. M., Reis, L. P., Lau, N.: Adapted Control Methods for Cerebral Palsy Users of an Intelligent Wheelchair Manual, Special Issue on Autonomous Robot Systems, Journal of Intelligent and Robotic Systems, Springer, ISSN: 1573-0409, (2014).
7. Faria, B. M., Reis, L. P., Lau, N.: Cerebral Palsy EEG signals Classification: Facial Expressions and Thoughts for Driving an Intelligent Wheelchair, IEEE Int. Conf. Data Mining 2012, Biological D.M. Appl. in Healthcare Works, Bruxelas, page 33-40, (2012).
8. Faria, B.M., Vasconcelos, S., Reis, L.P., Lau, N.: A Methodology for Creating Intelligent Wheelchair Users' Profiles, International Conference on Agents and Artificial Intelligence, 6 (8): 171-179, February, (2012).
9. Sasaki, Y., Fellow, R.: The truth of the F-measure, Manchester: MIB-School of Computer Science, University of Manchester (2007).
10. Coppin, B.: Artificial Intelligence Illuminated, Canada: Jones & Bartlett Learning (2004).
11. Kirkpatrick, S., Gelatt, C.D., Vecchi, M. P.: Optimization by simulated annealing, Science, 220 (4598): 671-680 (1983).
12. Glover, F.: Tabu search (Part I), ORSA Journal on Computing, 1: 190-206, (1989).
13. Holland, J. H.: Adaptation in natural and artificial systems, Univ. Michigan Press (1975).

14. Palisano, R. J., Rosenbaum, P., Bartlett, D., Livingston, M. H.: Content validity of the expanded and revised Gross Motor Function Classification System, D. Medicine and Child Neurology, 50 (10): 744-750 (2008).