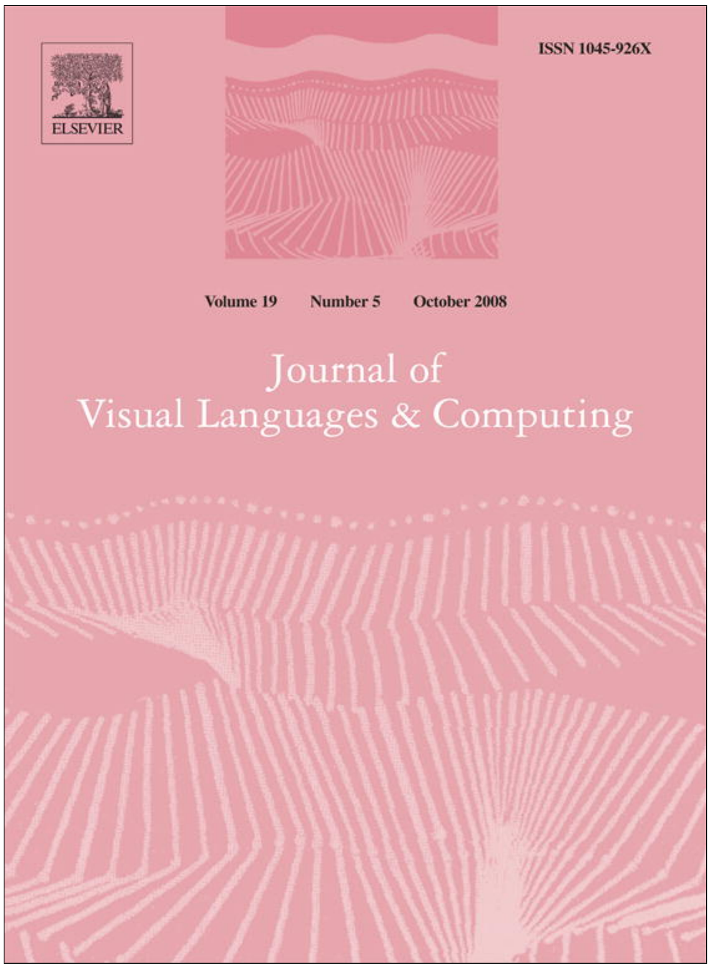

ISSN 1045-926X
ELSEVIER
Volume 19 Number 5 October 2008
Journal of
Visual Languages & Computing

Available online at www.sciencedirect.com
ScienceDirect
Journal of Visual Languages and Computing 19 (2008) 523–538
Journal of Visual Languages & Computing
www.elsevier.com/locate/jvlc

Proposing a speech to gesture translation architecture for Spanish deaf people

R. San-Segundo*, J.M. Montero, J. Macías-Guarasa, R. Córdoba, J. Ferreiros, J.M. Pardo

Grupo de Tecnología del Habla, Dpto. Ingeniería Electrónica, E.T.S.I. Telecomunicación, UPM, Ciudad Universitaria sn, 2804, Madrid, Spain

Received 10 January 2005; received in revised form 16 June 2007; accepted 20 June 2007

Abstract

This article describes an architecture for translating speech into Spanish Sign Language (SSL). The architecture proposed is made up of four modules: speech recognizer, semantic analysis, gesture sequence generation and gesture playing. For the speech recognizer and the semantic analysis modules, we use software developed by IBM and CSLR (Center for Spoken Language Research at University of Colorado), respectively. Gesture sequence generation and gesture animation are the modules on which we have focused our main effort. Gesture sequence generation uses semantic concepts (obtained from the semantic analysis) associating them with several SSL gestures. This association is carried out based on a number of generation rules. For gesture animation, we have developed an animated agent (virtual representation of a human person) and a strategy for reducing the effort in gesture animation. This strategy consists of making the system automatically generate all agent positions necessary for the gesture animation. In this process, the system uses a few main agent positions (two or three per second) and some interpolation strategies, both issues previously generated by the service developer (the person who adapts the architecture proposed in this paper to a specific domain). Related to this module, we propose a distance between agent positions and a measure of gesture complexity. This measure can be used to analyze the gesture perception versus its complexity. With the architecture proposed, we are not trying to build a domain independent translator but a system able to translate speech utterances into gesture sequences in a restricted domain: railway, flights or weather information.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Animation; Animated agents; Lifelike characters; Spanish sign language; Speech to gesture translation; Gesture complexity

1. Introduction

Speech and language technologies have always had an important relationship with their corresponding animated agents (virtual representations of a human person). These technologies provide them with new capabilities that improve the interface between animated agents and the end users (people who interact with the system to obtain some

*Corresponding author. Tel.: +34 91549 5700; fax: +34 91336 7323.
E-mail addresses: lapiz@die.upm.es (R. San-Segundo), juancho@die.upm.es (J.M. Montero), macias@die.upm.es (J. Macías-Guarasa), cordoba@die.upm.es (R. Córdoba), jfl@die.upm.es (J. Ferreiros), pardo@die.upm.es (J.M. Pardo).

1045-926X/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2007.06.002

service). The users can interact with animated agents using the common language. A community of scientists worldwide is developing and evaluating virtual humans embedded in spoken language systems. These systems provide a great variety of services in very different scenarios. Some researchers have embedded animated agents in information kiosks in public places [1]. At HTK in Stockholm, Joakim Gustafson [2], Björn Granström [3] and their colleagues have developed several multimodal dialogue systems where animated agents were incorporated to improve the interface. These include Waxholm [4] (a travel planning system for ferryboats in the Stockholm archipelago), August [5] (an information system at the Culture Center in Stockholm), and AdApt [6] (a mixed-initiative spoken dialogue system, in which users converse with a virtual real estate agent to locate apartments in Stockholm).

Education is another domain in which language and animated agent technologies can be combined. At this point, it is necessary to remark that there is a CSLU Toolkit which integrates an animated agent named Baldi. This toolkit has been developed at CSLU (Center of Spoken Language and Understanding, Oregon Graduate Institute, OGI) [7,8] which is now being expanded at CSLR (Center for Spoken Language Research at University of Colorado) [9]. This toolkit facilitates the speedy development of interactive books with multimedia resources and natural interaction. Nowadays, researchers have generated systems and architectures for representing and managing behaviors of animated agents. In [10,11], the authors present a good overview of computational models for developing believable virtual humans.

Not only research centers but also companies like Microsoft and IBM are interested in animated agents. Microsoft has developed a software platform [12] where users can use several animated agents or create new ones (http://www.microsoft.com/msagent/). This platform began with the Persona Project [13]. IBM is also interested in technology which will be the future for human–computer interfaces. In both the aforementioned systems, the synergy between language and virtual agent technologies is due to the fact that virtual humans offer a friendlier computer–user interface. This synergy becomes stronger in our case where we want to develop a system to translate speech into gestures for Deaf people. In the recent years several groups have shown interest in machine translation for Sign Languages, developing several prototypes based on different language translation techniques: example-based [14], rule-based [15], full sentence [16] or statistical [17] approaches. In a speech to sign language translation system, the virtual agent appears as an essential part of the system. It represents the gestures obtained from the semantic analysis of the recognized words. All of the aforementioned agent platforms suffer the inconvenience of the great effort needed to develop the agent animations. This is one of the problems we focus on in this paper: the development of a platform where minimal effort is required to create a new agent animation. This is an important aspect because the amount of gestures required by our system is higher than those of the aforementioned systems.

Sign Languages vary greatly depending on the country and even between different regions within the country. In 1960, Professor William Stokoe [18] presented the first conclusions from several studies on ASL (American Sign Language). After these studies, new works appeared not only in the USA [19,20] but also in Europe [21–23], Africa [24] and Japan [25]. In Spain, during the last twenty years, there have been several proposals for normalizing Spanish Sign Language (SSL), but none of them has been very well received by the Deaf community. These proposals tend to constrain the sign language, limiting its flexibility. In 1991, MA. Rodríguez [26] carried out a detailed analysis of SSL illustrating the main characteristics. She detailed the differences between the sign language used by Deaf people and the standardization proposals. This work has been one of the main studies on SSL and it has been the main reference for our work. Because of variations in SSL (even between different regions in Spain), we have proposed a flexible architecture which is easy to modify. The system's behavior is defined in auxiliary text files also easily modified by system developer (the person who adapts the architecture proposed in this paper to a specific domain): a context-free grammar, generation rules and gesture animations (Fig. 1).

In Section 2, we present an overview of the architecture describing the principal modules. Section 3 describes the speech recognizer. Section 4 presents the Phoenix parser. In Section 5, the gesture sequence generation module is described. Section 6 shows the gesture-playing module and the tools needed to generate the agent animations. Finally, Section 7 summarizes the main conclusions

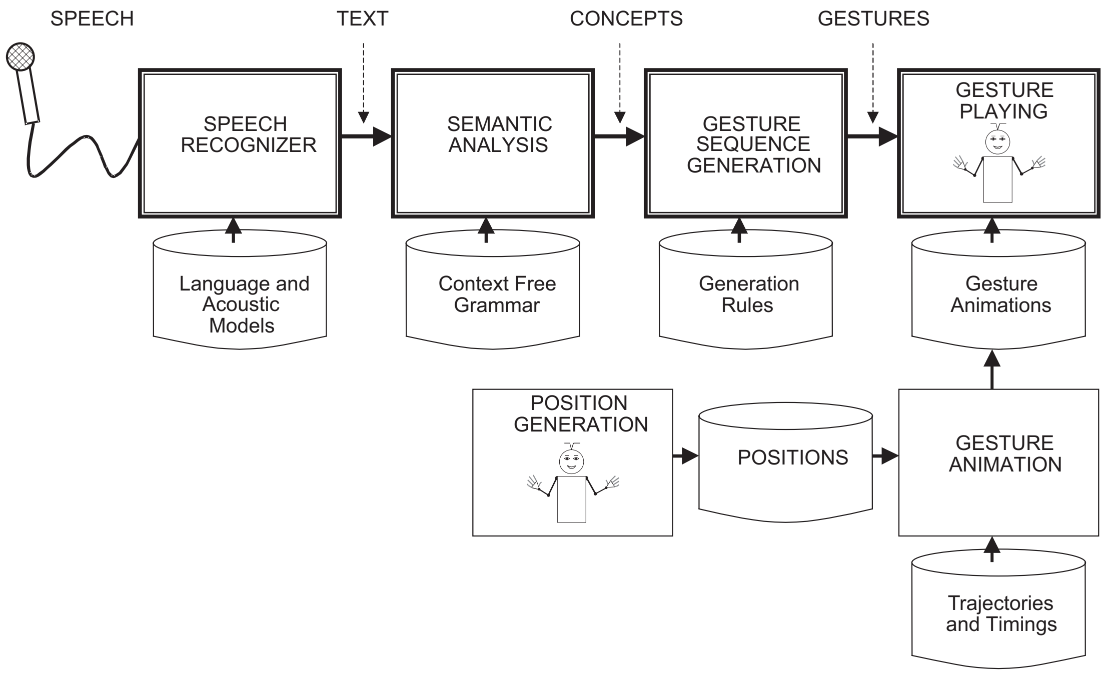


Fig. 1. Speech to gesture translation architecture.

of the work and Section 8 presents several avenues of further research.

2. System overview

In Fig. 1, we show the architecture proposed for translating speech into gestures for Deaf people. In this diagram, we have remarked on the four main modules, which carry out the four steps needed in the translation process: speech recognition, semantic analysis, gesture sequence generation and gesture playing. The position generation and gesture animation modules permit the animations needed by the gesture-playing module to be generated.

The first module (speech recognition) converts speech utterances into text words. For this module, we have used the latest version of the IBM ViaVoice software for Spanish. It is a voice recognition product that includes essential dictation, and command/control features. This module uses language and acoustic models adapted to Spanish pronunciation.

The semantic analysis module carries out a semantic evaluation of the text sentence (output from the speech recognizer) extracting the main concepts related to the application domain. For this module, we have used the Phoenix v3.0 parser developed at CSLR. This parser uses a context-free grammar to extract the semantic concepts from the word sequence.

The gesture sequence generation module processes the semantic analysis output and assigns a sequence of gestures to the semantic concepts. In this process, we consider four situations: one concept is mapped onto a unique gesture, one concept generates several gestures, some concepts are mapped onto a unique gesture, and finally, several concepts generate several gestures. We have studied different analyses of SSL [26,27] and we propose solutions for the four aforementioned situations. To resolve these situations, we consider both the Context-Free Grammar (semantic analysis module) and the Generation rules (gesture sequence generation module). The semantic analysis and the gesture sequence generation modules are designed for restricted domain services, i.e. the Context-Free Grammar and the Generation rules used in these modules do not contain all the possibilities for any interacting context. When the number of interacting contexts grow, the system complexity increases causing a drop in performance. Because of this, these modules must be adapted to a specific domain like railway, flight or weather information in order to guarantee a good performance. In the same way, although the IBM speech recognizer can be used for a wide variety of contexts, the performance increases greatly when we adapt its language and acoustic models to the application domain and to the speaker.

In the fourth module, an animated agent signs the gesture sequence. This agent is a very simple representation of a human being but it permits the gestures of the sign language to be represented properly. For each gesture, the system plays a different agent animation (all gesture animations must be created previously). Some companies and research centers have developed animated agents for human–computer interfaces. The main problem with these agents is the great effort needed to create an animation. Each animation needs at least 20 agent positions per second to guarantee a continuous movement. In this paper, we propose a strategy for generating animations with minimal effort. In our proposal, the system automatically creates a significant percentage of agent positions needed to generate the animation. The system developer only defines a few positions manually (main positions) and several interpolation strategies (trajectories and timings). The system uses these main positions (two or three per second) and the interpolation strategies to create all agent positions needed in the continuous movement.

3. Speech recognizer

For this module we have used the latest version of the IBM [28] ViaVoice software for Spanish: the release 8 Standard Edition [29]. IBM ViaVoice Standard Edition is a voice recognition product that includes essential dictation, and command/control features. This software permits users to dictate, edit, correct, and format text in a speech-enabled word processor, SpeakPad or directly into Microsoft Word 2002, Word 2000, and Word 97 to create letters, reports, and other documents easily. With this software, it is also possible to control applications using the voice. In our case, we use IBM ViaVoice for controlling our program (command features) and for dictating the sentences that will be translated into gestures (dictation features). The main features of the IBM ViaVoice 8 Standard Edition are:

- It comes with a 100,000 word basic vocabulary that can be customized to add new words, addresses, acronyms, and other personal phrases and expressions. The developer can increase this vocabulary. IBM ViaVoice can manage up to 164,000 active words working in real time.
- It knows the orthography and pronunciation of 475,000 words.
- The recognition engine incorporates a language model capable of distinguishing homophones.
- This speech recognition software permits user adaptation of both the acoustic and language models. Several user profiles are possible on the same PC.
- IBM ViaVoice permits dictation voice shortcuts to be created to insert blocks of text (e.g. greetings, addresses, salutations and quotes) directly into your dictation documents.
- It incorporates noise models and background noise adaptation to deal with noise produced by the user (breath, lip smack, tongue click, etc.) and the environment.

As previously stated, the translation architecture proposed in this paper is oriented towards developing restricted domain applications such as travel or cinema ticket reservation. In these cases, the vocabulary used by the IBM recognizer must be adapted to these specific domains (eliminating useless words). Additionally, the acoustic and language models can be adapted to the speaker. These two facts allow us to obtain high word accuracy.

4. Semantic analysis

This step tries to extract the main semantic concepts from the text sentence (output from the speech recognizer). For this module, we have used the Phoenix v3.0 parser developed at CSLR by Wayne Ward [30–33]. The Phoenix parser is designed for the development of simple, robust Natural Language interfaces to spoken language applications. Often, spontaneous speech utterances are ill formed causing the recognizer to make recognition errors. Because of this, the parser needs to be robust in order to deal with errors in recognition.

Phoenix parses each input utterance into a sequence of one or more semantic frames. The system developer (the person who adapts the architecture proposed in this paper to a specific domain) must define a set of frames and provide grammar rules that specify the word strings that can fill each slot in a frame. This information is used by the parser engine to map input word strings onto a sequence of semantic frames. A Phoenix frame is a named set of slots where the slots represent related pieces of information. A frame represents some basic type of action or object for the final speech to

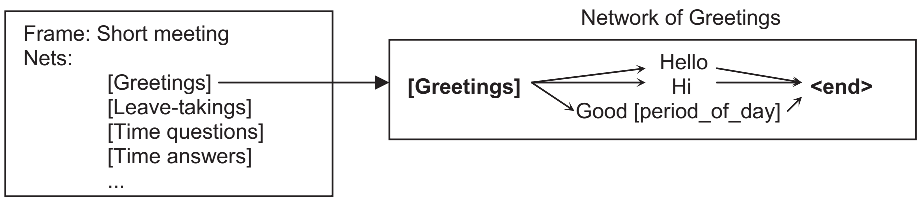


Fig. 2. Example frame for short meetings.

```
Input:   Good afternoon sir, could you tell me the time?, please

Parse:   Frame: Short meetings
                 [Greetings](good [period_of_day](afternoon))
                 [Time questions](tell me [time](the time))
```

Fig. 3. Example-parsed output.

sign language application. Fig. 2 shows an example frame for short meetings. Slots in a frame represent information that is relevant to the action or object. Each slot has an associated Context-Free Grammar that specifies word string patterns that match the slot (grammar rules or nets). The grammar rules are compiled into Recursive Transition Networks (RTNs) and the slot name will be the root of the corresponding semantic parse tree. In this parser, it is possible to define new slots within other slots implementing a hierarchical structure. An example-parsed frame is shown in Fig. 3.

The parsing process is a dynamic programming algorithm where grammars for slots are matched against a word string to produce a slot graph. The set of active frames defines a set of active slots. Each slot points to the root of an associated RTN. These networks are matched against the input word sequence by a top-down RTN chart-parsing algorithm. The parser proceeds from left to right in an attempt to match each slot network starting with each word of the input, as in

```
For (each word of input)
For (each active slot)
match_net (slot, word)
```

The function match_net is a recursive function that matches an RTN against a word string beginning at the specified word position. The function produces all matches for the network starting at the word position and may have several different endpoints. The networks are not designed to parse full sentences, just sequences of words. Because of this, the parser is very robust for use with automatic speech recognizers. The RTNs for the slots call other nets in the matching process. Each time a net match is attempted (all nets, not just slots), this is noted in the chart. All matched networks are added to the chart as they are found. Any time a net match is attempted, the chart is first checked to see if the match has been attempted before. When a slot match is found, it is added to the slot graph. Each sequence of slots in the slot graph is a path. The score or metric for the path is the number of words accounted for by the sequence. Words are not skipped in matching a slot, but words can be skipped between the matched slots. The graph growing process prunes poor scoring paths. The pruning criteria are firstly the number of words accounted for and second, the degree of fragmentation of the sequence. If two paths cover the same portion of the input and one accounts for more words than the other, the less complete is pruned. If the two paths account for the same number of words, and one uses fewer slots than the other, the one with more slots is pruned. The resulting graph represents all of the sequences found that have a score equal to the best. The sequences of slots represented by the graph are then grouped into frames. This is done simply by assigning frame labels to the slots. Again in this grouping, the least fragmented parse is preferred. For example, if two parses each have five slots, and one uses two frames and the other uses three, then the parse using two frames is preferred. The result is a graph of slots, each labeled with one or more frame labels. Each path through the graph, scoring equally, is a parse. This mechanism naturally produces partial or fragmented parses. The dynamic programming

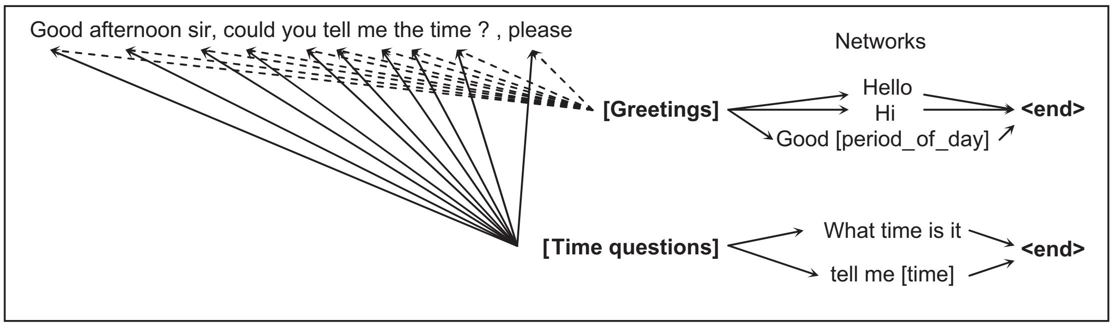


Fig. 4. The matching process.

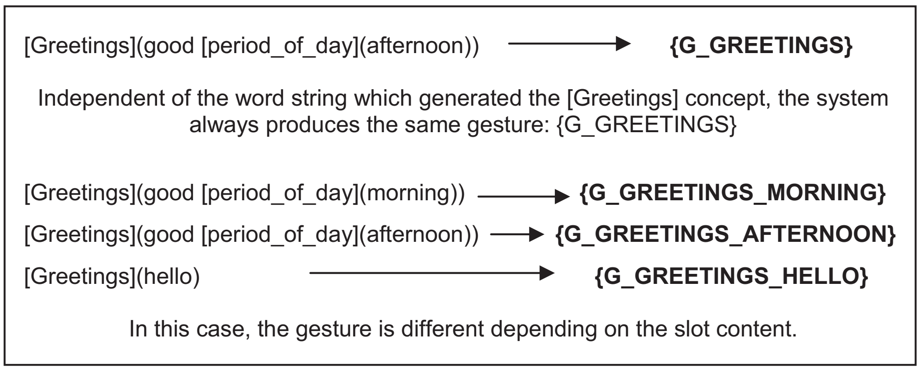

[Greetings](good [period_of_day](afternoon))
{G_GREETINGS}
Independent of the word string which generated the [Greetings] concept, the system always produces the same gesture: {G_GREETINGS}
[Greetings](good [period_of_day](morning))
{G_GREETINGS_MORNING}
[Greetings](good [period_of_day](afternoon))
{G_GREETINGS_AFTERNOON}
[Greetings](hello)
{G_GREETINGS_HELLO}
In this case, the gesture is different depending on the slot content.

Fig. 5. Assigning a unique gesture to a semantic concept.

search produces the most complete, least fragmented parse possible, given the grammar and the input Fig. 4.

The parser segments the input at natural breakpoints and carries out garbage collection at its data structures. This allows it to parse entire reports as single utterances if necessary. The processing speed is generally linear with the length of the input. More details on the parser can be consulted in [32].

5. Gesture sequence generation from semantic analysis

The semantic analysis output is a sequence of parsed slots: each parsed slot is considered a semantic concept. In this step, the gesture sequence generation processes the semantic analysis output to obtain the final gestures that the animated agent will sign. In this process, we describe four situations:

5.1. *One semantic concept corresponds to a specific gesture*

In this case, a semantic concept (parsed slot) is directly mapped onto a specific gesture. The translation is simple and it consists of assigning one gesture to each semantic concept. This gesture can be a default translation, independent of the word string, or can be different depending on the word string from which it is generated (Fig. 5).

5.2. *Several semantic concepts are mapped onto a unique gesture*

The second situation appears when several concepts generate a unique gesture. This situation should be solved in the previous step (semantic analysis). The solution is to unify the concepts (slots) in the parser grammar (resulting in just one concept or slot) and to proceed as in the previous situation (Fig. 6).

The Phoenix parser (semantic analysis) provides the possibility of organizing the concepts (slots) into a hierarchical structure. This fact allows us to establish more complicated relationships between them in order to generate a unique gesture. As in the previous situation, the gesture being generated may or may not differ according to the slot content.

[Asking](tell me) [Time] (the time) → **{G_ASK-TIME}**

Two concepts ([Asking] and [Time]) together generate a unique gesture. To solve the problem, we unify both slots in the parser grammar and we proceed as in the previous situation.

[AskingTime] (tell me the time) → **{G_ASK-TIME}**

Fig. 6. Assigning a unique gesture to several semantic concepts.

ACTION TENSE

I played football yesterday **{G_PAST}** {G_I} **{G_PLAY}** {G_FOOTBALL} {G_DATE_YESTERDAY}

[Subject](I) → **{G_I}**
[Play](played) → **{G_PAST} {G_PLAY}**
[Football](football) → **{G_FOOTBALL}**
[Date](yesterday) → **{G_DATE_YESTERDAY}**

In this example, the verb generates 2 gestures: action and tense. The tense gesture must be introduced at the beginning of the gesture sequence. The sign language distinguishes 3 verb tense: past, present and future. The default tense is present and it does not need to be assigned. In the other cases, it is necessary to introduce a tense gesture at the beginning of the sentence.

SUBJECT

In Spanish (as opposed to English), it is quite common to omit the subject of the verb. This fact does not cause any ambiguity because the verb conjugation is different depending on the action subject. In these cases, the verb concept must generate 3 gestures: term, subject and action. In the previous example, we could omit the subject in Spanish:

Jugué al fútbol ayer. (I played football yesterday)

Jugué al fútbol ayer **{G_PAST} {G_I} {G_PLAY}** {G_FOOTBALL} {G_DATE_YESTERDAY}

[Play](jugué) → **{G_PAST} {G_I} {G_PLAY}**
[Football](fútbol) → **{G_FOOTBALL}**
[Date](ayer) → **{G_DATE_YESTERDAY}**

GERUND

For indicating that the action is (or was) in process, the gesture associated with the verb action is repeated twice.

I was playing football when you arrived

{G_PAST} {G_I} **{G_PLAY} {G_PLAY}** {G_FOOTBALL} {G_PAST} {G_YOU} {G_ARRIVE}

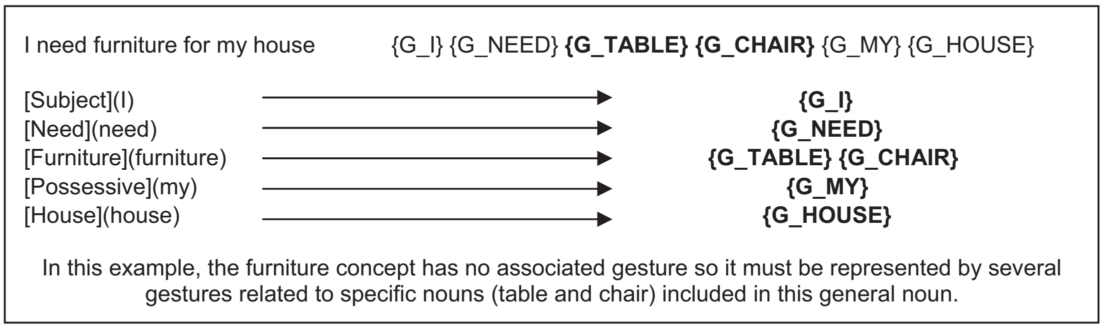
[Subject](I) → **{G_I}**
[Play](played) → **{G_PAST} {G_PLAY} {G_PLAY}**
[Football](football) → **{G_FOOTBALL}**
[Subject](you) → **{G_YOU}**
[Arrive](arrived) → **{G_PAST} {G_ARRIVE}**

Fig. 7. Type of gesture sequences generated by verb concepts.

5.3. *One semantic concept generates several gestures*

The third situation occurs when it is necessary to generate several gestures from a unique concept. This problem strongly justifies the need for the gesture sequence generation module. Similar to previous sections, the gesture sequence and its order can depend on the concept and its content, or just on the concept. This situation appears in many translation issues:

- Verbs. A verb concept generates a gesture related to the action proposed by the verb and some gestures provide information about the action tense (past, present or future), the action subject and gerund action (Fig. 7).



I need furniture for my house {G_I} {G_NEED} **{G_TABLE} {G_CHAIR}** {G_MY} {G_HOUSE}

[Subject](I)	**{G_I}**
[Need](need)	**{G_NEED}**
[Furniture](furniture)	**{G_TABLE} {G_CHAIR}**
[Possessive](my)	**{G_MY}**
[House](house)	**{G_HOUSE}**

In this example, the furniture concept has no associated gesture so it must be represented by several gestures related to specific nouns (table and chair) included in this general noun.

Fig. 8. Gestures for general nouns not presented in the sign language.

[Clay]	**{G_LAND} {G_WITH} {G_WATER}**
[Quarry]	**{G_STONE} {G_EXCAVATE} {G_MOUNTAIN}**
[Burrow]	**{G_HOLE} {G_EXACTLY} {G_RABBIT} {G_HOME}**
[Fence]	**{G_GREEN} {G_DEFENSE}**

Fig. 9. Examples of Lexical–Visual Paraphrases.

- General and Specific Nouns. In sign language there is a tendency to refer to objects with high precision or concretion. As a result of this, there are numerous domains where several specific nouns exist, but there is no general noun to refer to them collectively. For example, this happens with metals: there are different gestures to refer to gold, silver, copper, etc. but there is no general gesture to refer to the concept of metal. The same thing happens when considering furniture: there are several gestures for table, chair, bed, etc. but there is no general gesture to refer to the concept of furniture. This problem is solved in sign language by introducing several specific gestures (Fig. 8).
- Lexical–Visual Paraphrases. Frequently, new concepts (in Spanish) appear and they do not correspond to any gesture in sign language. In order to solve this problem, Deaf people use paraphrases to represent a new concept with a sequence of known gestures. This solution is the first step for representing a new concept. If this concept begins to appear frequently, the gesture sequence will be replaced by a new gesture for reducing the representation time. Some examples of Lexical–Visual Paraphrases are shown in Fig. 9.
- Complex Signs. Similar to the paraphrases, complex signs are made up of several gestures. Each of theses gestures can be used independently but they are represented together in order to show another concept. Some examples are shown in Fig. 10.

[Week-end]	**{G_SATURDAY} {G_SUNDAY}**
[Hospital]	**{G_PATIENT} {G_HOUSE}**
[Democracy]	**{G_VOTE} {G_LIBERTY}**

Fig. 10. Examples of Complex Signs.

- The gestures are language representations which are more difficult to memorize and distinguish than words. Because of this, the gesture dictionary is smaller than the Spanish word dictionary. This fact makes it necessary to combine gestures (complex signs) in order to represent other concepts.
- Date and Time. As it is shown in Fig. 11, a date representation can be made with one or several gestures. The time generally requires several gestures for a full representation.
- Emphasis. When you want to emphasize some aspect of a sentence, the way of doing so is by repeating the associated gesture. For example, if we want to emphasize the possessive "my" in the sentence "this is my house", we should repeat the associated gesture. Nowadays, the commercial speech recognizers do not detect emphasis or emotion in the speech, so this aspect cannot be translated into sign language. We expect that this

[Date](tomorrow)	**{G_TOMORROW}**
[Date](May 3rd, 2001)	**{G_MAY} {G_THIRD} {G_TWO} {G_THOUSAND} {G_ONE}**
[Time](4:35)	**{G_HOUR} {G_FOUR} {G_AND} {G_THIRTY} {G_FIVE}**
[Time](6:45)	**{G_HOUR} {G_SEVEN} {G_QUARTER_TO}**

Fig. 11. Dates and Times examples.

[House](houses)	**{G_HOUSE} {G_HOUSE}**
[Apple](apples)	**{G_SEVERAL} {G_APPLE}**
[Car](cars)	**{G_CAR_2HANDS}**

Fig. 12. Plural noun examples.

characteristic can be available in the near future.

- Plural Nouns. There are several ways of specifying an object in plural (all of them with the same meaning): repeating the gesture, introducing an adverbial gesture or representing the gesture with both hands. In Fig. 12, we show several examples.
- Gender. A new gesture can be introduced into the sequence to indicate the gender of some object. Generally the gender can be deduced by context and it is not necessary to specify it. This gesture appears when the gender is necessary for the meaning or we want to remark on this fact.

5.4. Several semantic concepts generate several gestures

Finally the most complicated situation appears when it is necessary to generate several gestures from several concepts with certain relationships between them. Some examples are the followings:

- Verb/action gesture depending on the subject of the action. For example, the verb "fly" is represented with different gestures depending on the subject of the action: bird, plane, butterfly, etc.
- A similar situation crops up when the gesture associated when an adjective changes depending on the qualified object. For example, the gesture for the adjective "good" is different when referring to a person or a material thing.

These cases, presented in this section, are less frequent than those presented in Sections 5.1–5.3. In our system, the cases presented in this section are solved by mixing the strategies carried out in Sections 5.2 and 5.3. First, we group the different concepts under a unique concept structure, and then we apply similar strategies as in Section 5.3, to generate a gesture sequence from a unique semantic concept structure. The characteristics of the sign language used by Spanish people have been extracted from [26] where we obtained an extended and detailed description.

6. Gesture animation

In order to represent the gesture sequence (generated in the previous module), we have developed an animated agent. This agent is a simple representation of a human person but it is detailed enough to represent the gestures used in sign language. In this section, we focus on the description of this agent and gesture design. Around the world, several companies and research centers have developed animated agents for human–computer interfaces. The main problem with these agents is the great effort needed to build an animation: it is necessary to generate several agent positions per second in order to obtain a continuous movement. One of the main issues dealt with in this section is the way to generate gesture animations from a very small number of agent positions. The main target for us has been to reduce drastically the effort in gesture design.

6.1. The animated agent: AGR (agent for gesture representation)

For representing the gestures, we have developed a very simple animated agent. This agent is made up by combining rectangles, circles and different sized lines (Fig. 13).

The AGR is made up of five fixed points (the center of the circular head and the four points of the rectangular torso) and 60 mobile points: 18 for the right arm, hand and fingers, 18 for the left arm,

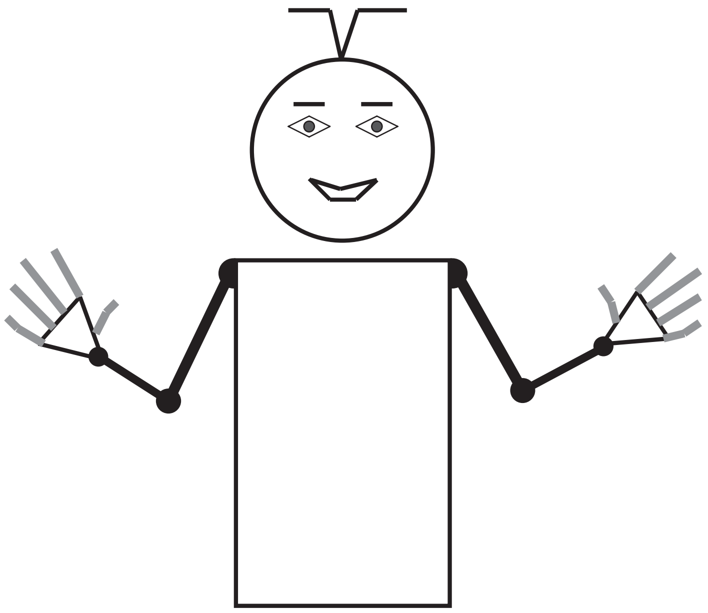


Fig. 13. AGR: agent for gesture representation.

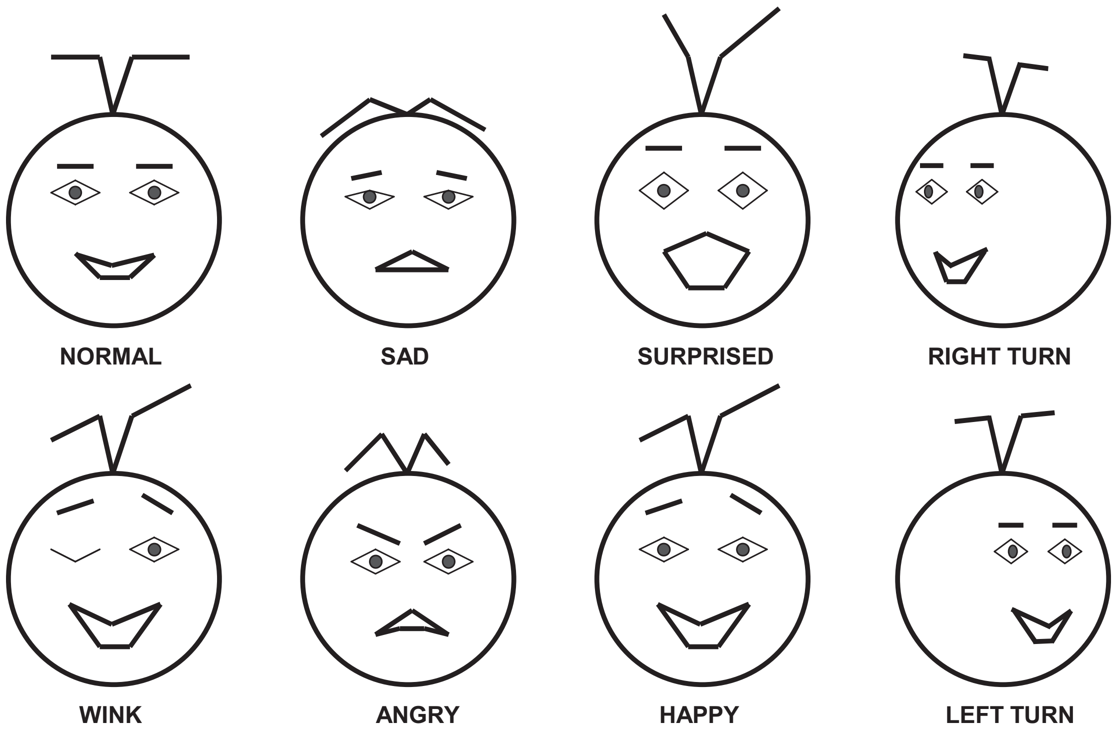


Fig. 14. Facial expressions.

hand and fingers, and 24 for the face representation (eyes, mouth, eyebrows and two hairs).

6.2. *AGR's head and face*

The AGR's head is represented by several lines compiled from 24 points and three circumferences: two for the pupils and another for the outline of the head. Each eye is made up of four points connected by four lines, and each eyebrow is a single line. The mouth is drawn using five points and five lines (two for the upper lip and three for the lower lip). The two hairs are represented by three points and two lines each (with the same point of origin). Although, they are not necessary, they help to reinforce the facial expression. The architecture has the possibility to hide the hair. This possibility is configurable by the system developer. In Fig. 14, we show different facial expressions.

In order to simulate a head turn, it is necessary to change the position (inside the head) and the size of eyes, eyebrows and mouth (Fig. 14).

6.3. AGR's arm and hand

For the arm and hand representation, we use 18 points and 15 lines. Two points represent the arm: shoulder and elbow. For the hand, we use 16 points: one for the wrist, two additional points for the palm of the hand and 15 for the fingers (knuckle, phalanx and tip). The forefinger and little finger knuckles coincide at the corners of the palms (Fig. 15). In order to distinguish the front (palm) and the back of the hand, we have incorporated the possibility of changing the color of the hand: white to refer to the front of the hand (palm) and gray to refer to the back (Fig. 15).

The finger lines are drawn in a lighter shade of gray in order to identify their movements easily. Although the color of the back of the hand is the same as the color of the fingers, it is not a problem. In a normal movement, fingers are never bent over the back of the hand.

Three articulating points per finger (knuckle, phalanx and tip) are enough to represent clearly the gestures in sign language. In Fig. 16, we show the hand letter positions (dactylography).

In order to specify an agent position, we have developed a new tool where the system developer can modify and set up the AGR position: face, arm and hand positions. This tool has the following characteristics:

- For each agent point or agent region, the system developer can define different representation planes (from zero to six). These planes represent the distance from the view point: six is the closest plane and zero the furthest away plane. This utility permits the developer to specify which lines are drawn first and which should be drawn subsequently. The tool begins drawing the lines and points associated with the lower representation plane. By default, when all the points are in the same representation plane, the program draws the agent parts in the following order: head, torso, right arm, right hand, left arm and finally, the left hand (wrist, palm and fingers).
- The head, torso and palms are opaque regions. If another body part is in the same position as any of these regions, and it has a lower representation plane, it will not appear in the drawing. That region will hide this part of the body.
- The position tool permits the developer to store/recover any definition of agent position in/from a file. This is an important feature for the generation of the agent animation (see the next section).
- In order to generate agent positions faster, the tool offers the possibility to store/recover several parts of the agent independently: head and right or left hands. It is possible to combine different face expressions with several hand gestures.
- Furthermore, the tool provides a zoom utility to facilitate the design of details.

When specifying the agent position, the program establishes limitations concerning the length of some parts: arm, fingers, eyebrow, etc. The target is to avoid generating extremely deformed gestures. It also checks some conditions, e.g. that eyes, eyebrow and mouth are within the head limits, or that the pupil is situated within the eye limits.

6.4. Obtaining gesture animations from agent positions

An animation is generated automatically from a very small set of agent positions. These are defined in advance using the tool described in the previous section. The main target of this module is to generate an animation using as few positions as

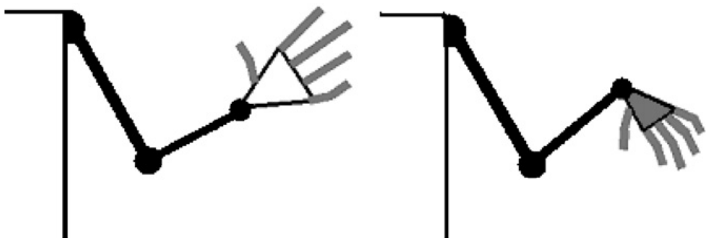


Fig. 15. Arm and hand detail.

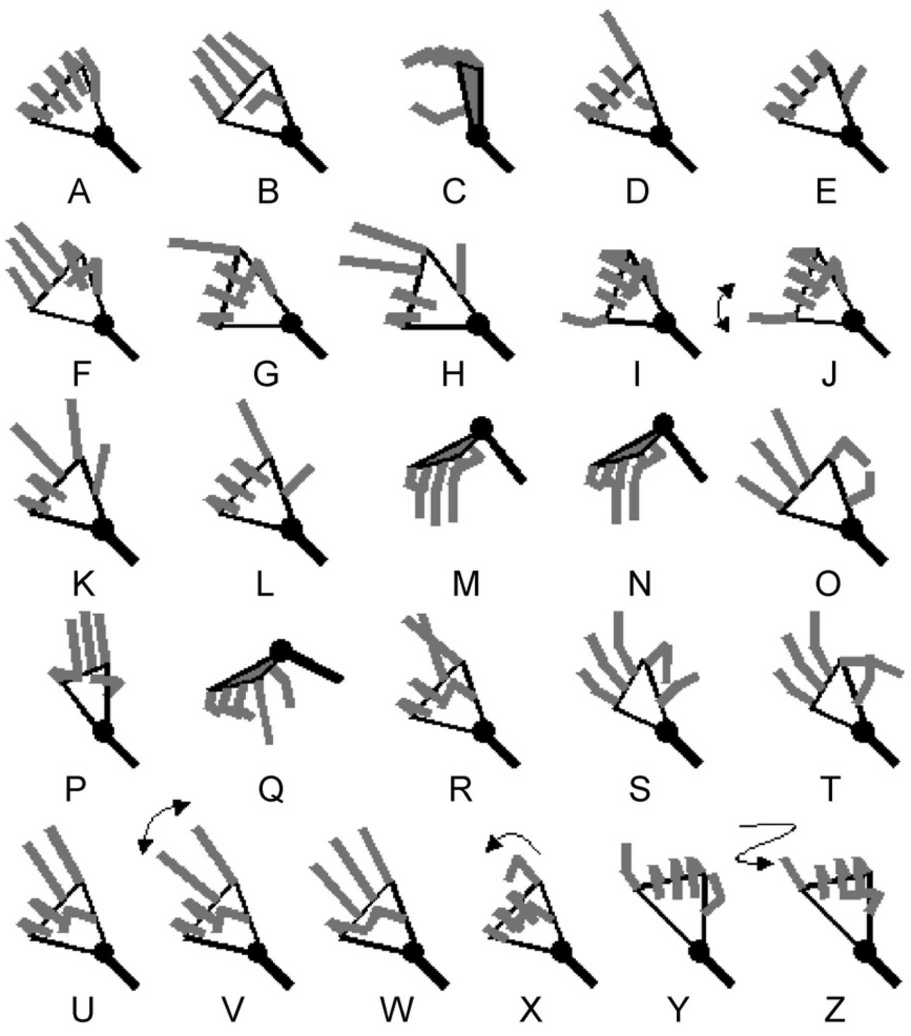


Fig. 16. Signs for the letters.

possible in order to reduce drastically the effort of generating gesture animations. A typical gesture takes approximately 2 s. This means that, considering 20 frames per second (for a continuous movement), we should create 40 frames/agent positions for a typical gesture. In sign language, there are more than 5000 different gestures. Animation creation is no trivial task. In this paper, we propose a strategy that reduces this effort. The main idea is to define a small number of frames/positions (around four or five per gesture), and to generate the intermediate positions automatically. The program creates these frames by interpolation. For any subsequence (positions created automatically between two positions defined by the developer), the system developer can specify different interpolations. In order to design an interpolation, it is necessary to define two aspects: the trajectory and timing.

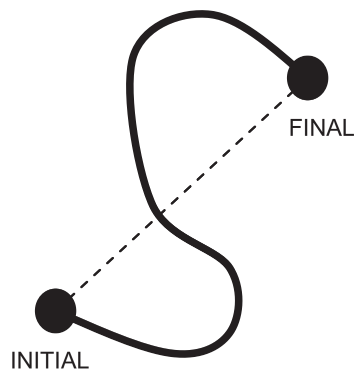


Fig. 17. Trajectory specification.

6.5. *Trajectory specification*

The developer can define the trajectory that any point of the agent body will follow when moving from the initial to the final position (see Fig. 17).

The trajectory is specified by the developer in a visual interface (with an adequate zoom) by moving the mouse cursor.

When the developer defines a trajectory, this trajectory can be assigned to a unique mobile point, a set of mobile points, or to all mobile points. No trajectory is assigned to a static point of the agent. For the mobile points for which the developer does not define any trajectory, the program generates a rectilinear one by default. This allows a complete specification. The default trajectory is not fixed and can also be modified by the developer.

6.6. *Interpolation timing*

The second aspect to define is the timing: how fast the point passes through the different parts of the trajectory. The trajectory is a continuous line (infinite points) but the number of intermediate positions is small: around 10. Because of this, the developer needs to specify where, in the trajectory, the mobile point will be situated for each of the interpolated positions. In the same visual interface (Fig. 18), several intermediate circles appear, as many as there are intermediate positions. The developer can position each circle at any trajectory point (as it is not possible to change the circle order).

When the developer defines timing, it is associated with a unique mobile point, a set of mobile points or to all mobile points. Two points with the same trajectory can have different timings. By default, if no timing is specified, the program positions the intermediate points equidistantly. The interpolated positions/frames are created by the program combining the trajectory and timing

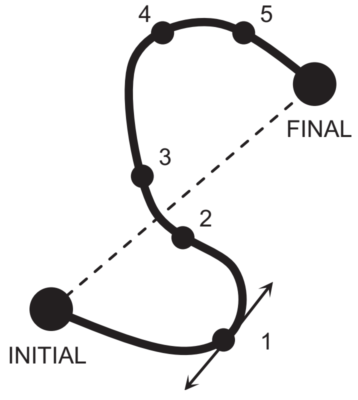


Fig. 18. Timing specification with five intermediate positions.

associated to each point. In this process, as in the position module, the program checks for limitations concerning the length of some parts of the body. The goal is to avoid generating extremely deformed gestures. It also checks certain conditions, e.g. that eyes, eyebrow and mouth must be within the head limits, or the pupil should be situated within the eye limits.

The gesture animation can be stored in a file; completely or partly (a sub sequence of frames). The subsequences are very useful for new gesture design.

6.7. *Playing a sequence of gestures*

As previously mentioned, a gesture animation is treated (and stored) as a set of agent positions (20 positions per second). When we want to play a sequence of gestures, we need to carry out two actions: first we concatenate the gesture animations in order to produce a continuous movement, and second, we define the speed of play. These two actions are described in the following sections.

6.8. *Gesture concatenation*

When concatenating two gestures, it is necessary to introduce new agent positions between the last position of a gesture and the first position of the next gesture. This action is very important in order to produce a continuous movement. One issue is to decide how many positions should be included between the two gestures. In this paper, we propose a variable number depending on the difference between the first and last positions of the consecutive gestures. The measuring of agent position difference, proposed in this paper, is the average Euclidean distance for all mobile points from one agent position to another: Eq. (1), where N is the number of mobile points and $P_{n\ Position\ X}$ is the mobile point n in agent position X:

$$Position\ Difference = \frac{\sum_{n=1}^{n=N} Euclidean\ Dist\ (P_{n\ position\ 2} - P_{n\ position\ 1})}{N}. \tag{1}$$

The greater the difference, the higher the number of positions should be and vice versa. The relationship between the Position Difference and the number of agent positions is given by Eq. (2), where PD is the Position Difference (see its representation in Fig. 19).

$$Number\ of\ Positions = 20 - 17e^{\alpha PD} \tag{2}$$

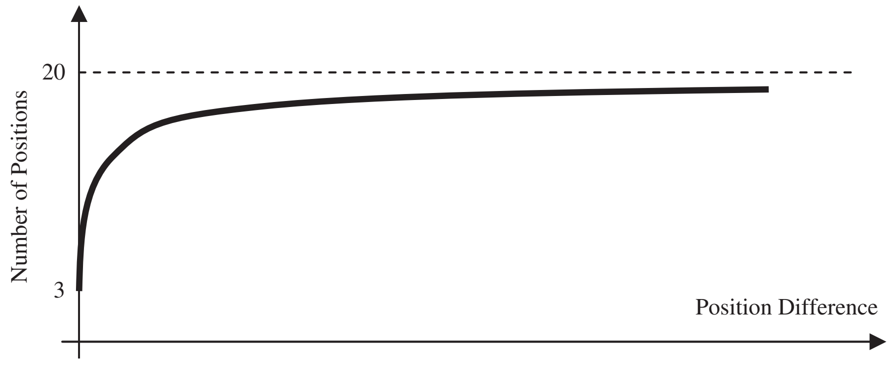


Fig. 19. Number of positions vs. position difference.

The α parameter can be modified. The target of this relationship is twofold: to guarantee a minimum number of positions for small differences and to define a limit of 20 intermediate positions, in order to avoid a long transition. The intermediate positions between two gestures are generated automatically by interpolation. The trajectory and the timing are the default strategies defined in the previous section: rectilinear trajectory and equidistant timing.

Expanding the ideas shown in this section, we propose a measure for the complexity of a gesture. This measure is the average Position Difference between consecutive agent positions throughout the gesture, Eq. (3), where M is the number of position in the gesture.

$$Gesture\ Complexity = \frac{\sum_{m=1}^{m=M-1} position\ Difference\,(m, m+1)}{M-1}. \tag{3}$$

The idea of proposing a gesture complexity measure is to define empirical measurements to compare them to how users perceive the gestures. This comparison permits the gesture to be modified according to these measurements. Greater gesture complexity could be associated with a more difficult gesture and it should be played more slowly. In further research, we plan to study the relationship between these measurements and gesture perception.

6.9. Synchronization

The final aspect we must keep in mind is the speed of the gesture sequence. This aspect is defined by the time between two consecutive agent position representations. If we want to slow the gesture, we increase this time and vice versa. Typically, there is a relationship between the duration of the utterance and the duration of the gesture sequence. The gesture sequence is around one and half and two times longer than the utterance duration. In some situations, the speaking rate (in phones per second) can vary significantly from one utterance to another. This variation can be also applied to the gesture sequence in two steps: first, we compute the percentage of increase or decrease in respect to the average speaking rate and then, this percentage is applied to the standard gesture sequence duration:

- If it is necessary to increase the gesture speed, the program automatically reduces the time between positions.
- On the other hand, if the program needs to reduce the speed, the time between positions is increased. In this case, there is a risk of loss of gesture continuity. When the number of positions per second is less than 15, the program interpolates new frames/positions using a rectilinear trajectory and an equidistant timing for the mobile points.

6.10. Gesture animation quality

In order to evaluate the quality of the gesture animation, we have performed a preliminary study about how Deaf people perceive gestures played by AGR. In this study 10 people (all of them knew SSL very well) have been asked to recognize several gestures played by the system. The system presents a gesture and the user has to recognize it. In order to evaluate the gesture animation, we have decided to consider the gestures corresponding to the letters (dactylography). This decision has been made based on two reasons:

- Using isolated gestures avoids the user recognizing the gesture using context information. This

kind of information can be very useful when recognizing gestures in a logical sequence.
- Secondly, letter gestures are very similar and they are a very good benchmark for evaluating the gesture animation quality.

The evaluation process was carried out in two steps:

- In the first step, 50 gestures were randomly selected and presented to every user. For every gesture, the user had to recognize the letter. At this point, more than 70% of the letters were correctly recognized. After this evaluation, the users were informed about which letters were recognized incorrectly.
- In a second step, another 50 gestures were presented to every user. In this case, almost 100% of the letters were recognized correctly. As the gesture is always played in the same way, the user learns easily how the system represents the gesture and recognizes it easily.

7. Conclusions

In this paper, we have proposed an architecture for a speech-to-gesture translator made up of four modules: speech recognizer, semantic analysis, gesture sequence generation and gesture sequence animation (gesture playing). The main effort in this work has focused on the gesture sequence generation and gesture animation. The gesture sequence generation is applied over the semantic analysis provided by the Phoenix parser. The most complex case is when a semantic concept generates several gestures. For this case, the detailed description of the SSL carried out by Rodríguez [26], has been very useful.

For the gesture animations, we have developed an animated agent and a strategy for reducing the gesture design time. This strategy consists of combining agent positions created by the developer and positions generated automatically by the system. The position generation is carried out by interpolation considering the previously designed point trajectories and timings.

In this work, we have also proposed a position distance metric and a measurement of gesture complexity. This measurement can be used to analyze the gesture perception versus its complexity. Although we have not carried out a complete evaluation as to how Deaf people perceive our agent gestures, preliminary evaluations with the letters of the alphabet reveal that Deaf people find less than 30% of the gestures difficult to understand. These situations occur only the first time the gesture is played. The subsequent times, as the gesture is always played in the same way, the Deaf person recognizes it easily.

With the architecture proposed, we do not want to build a domain independent translator but a system able to translate speech utterances into gesture sequences in a restricted domain: railway information, weather information, etc.

8. Future work

Three main ideas seem worthwhile for future research:

- The first is to evaluate in depth how Deaf people perceive the gestures played using our AGR agent. We will also try to establish a possible relationship between this perception and gesture complexity.
- The second is to carry out a more sophisticated method for gesture design. The idea is to develop a program able to generate all the positions of a gesture using several gesture characteristics: hand orientation, finger movement, articulation point, etc. For each characteristic, the developer will choose a value from a finite set.
- The last one is the possibility of collaborating with another research group working in gesture recognition in order to build a bi-directional system.

Acknowledgments

This work has been supported by the following projects TINA (UPM y DGUI-CAM. ref: R05/10922), ROBINT (MEC ref: DPI2004-07908-C02) and EDECAN (MEC ref: TIN2005-08660-C04). Authors also want to thank Mark Hallett for the English revision and the anonymous reviewers who helped with their comments to improve the quality and clarity of the presentation.

References

[1] J. Cassell, T. Stocky, T. Bickmore, Y. Gao, Y. Nakano, K. Ryokai, D. Tversky, C. Vaucelle Vilhjálmsson, MACK: Media lab Autonomous Conversational Kiosk, in: Proceedings of

Imagina: Intelligent Autonomous Agents, Monte Carlo, Monaco, 2002.

[2] J. Gustafson, Developing multimodal spoken dialogue systems- Empirical studies of spoken human-computer interactions, PhD. Dissertation, Department of Speech, Music and Hearing, Royal Institute of Technology, Stockholm, Sweden, 2002.

[3] B. Granström, D. House, J. Beskow, Speech and Gestures for Talking Faces in Conversational Dialogue Systems, Multimodality in Language and Speech Systems, Kluwer Academic Publishers, Donrecht, 2002 pp 209–241.

[4] J. Bertenstam, et al., The Waxholm system-A progress report, in: Proceedings on Spoken Dialogue Systems, Vigso, Denmark, 1995.

[5] M. Lundeberg, J. Beskow, Developing a 3D-agent for the August dialogue system, in: Proceedings on Audio–Visual Speech Processing, Santa Cruz, CA, 1999.

[6] J. Gustafson, L. Bell, Speech technology on trial: experiences from the August system, Journal of Natural Language Engineering: Special Issue on Best Practice in Spoken Dialogue Systems (2003) 273–286.

[7] S. Sutton, R. Cole, Universal speech tools: the CSLU toolkit, in: Proceedings of the International Conference on Spoken Language Processing, Sydney, Australia, 1998, pp. 3221–3224.

[8] R. Cole, et al., New tools for interactive speech and language training: using animated conversational agents in the classrooms of profoundly deaf children, in: Proceedings ESCA/SOCRATES Workshop on Method and Tool Innovations for Speech Science Education, London, 1999, pp. 45–52.

[9] R. Cole, S. Van Vuuren, B. Pellom, K. Hacioglu, J. Ma, J. Movellan, S. Schwartz, D. Wade-Stein, W. Ward, J. Yan, Perceptive animated interfaces: first steps toward a new paradigm for human computer interaction, IEEE Transactions on Multimedia: Special Issue on Human Computer Interaction 91 (9) (2003) 1391–1405.

[10] W.L. Johnson, J.W. Rickel, J.C. Lester, Animated pedagogical agents: face-to-face interaction in interactive learning environments, International Journal of artificial Intelligence in Education 11 (2000) 47–78.

[11] J. Gratch, J. Rickel, E. André, N. Badler, J. Cassell, E. Petajan, Creating interactive virtual humans: some assembly required, IEEE Intelligent Systems 17 (4) (2002) 54–63.

[12] Microsoft Agent web. 〈http://www.microsoft.com/msagent/index.html〉.

[13] G. Ball, D. Ling, D. Kurlander, J. Miller, D. Pugh, T. Skelly, A. Stankosky, D. Thiel, M. Van Dantzich, T. Wax, Lifelike Computer Characters: the Persona Project at Microsoft Research, 1999, 〈www.microsoft.com〉.

[14] S. Morrissey, A. Way, An example-based approach to translating sign language, in: Workshop Example-Based Machine Translation (MT X–05), Phuket, Thailand, September, 2005 pp. 109–116.

[15] M. Huenerfauth, A multi-path Architecture for Machine Translation of English Text into American Sign language animation, HLT-NAACL, Boston, MA, USA, 2004.

[16] S.J. Cox, M. Lincoln, J. Tryggvason, M. Nakisa, M. Wells, M. Tutt, S. Abbott, TESSA, A System to Aid Communication with Deaf People, ASSETS, Edinburgh, Scotland, 2002, p. 205–212.

[17] J. Bungeroth, H. Ney, Statistical sign language translation, in: Workshop on Representation and Processing of Sign Languages, LREC, 2004, pp. 105–108.

[18] W. Stokoe, Sign Language structure: an outline of the visual communication systems of the American deaf, Studies in Linguistics, Buffalo University Paper 8, 1960.

[19] L.B. Anderson, Aspect in Sign Language Morphology: The Role of Universal Semantics and Pragmatics in Determining Grammatical categories, Linguistics Research Laboratory, Gallaudet College (for the Symposium on Tense/Aspect: between semantics and pragmatics, UCLA, 4–6 May), 1979.

[20] C. Christopoulos, J. Bonvillian, Sign Language, Journal of Communication Disorders 18 (1985) 1–20.

[21] B. Hansen, Varieties in Danish Sign Language, Sign Language Studies 8 (1975) 249–256.

[22] J. Kyle, British Sign Language, Special Education 8 (1981) 19–23.

[23] B. Frokjaer-Jensen, The sciences of deaf signing, Copenhagen University 1980.

[24] C. Penn, R. Lewis, A. Greenstein, Sign Language in South Africa, South African Disorder of Communication 31 (1984) 6–11.

[25] M. Notoya, S. Suzuki, M. Furukawa, R. Umeda, Method and acquisition of sign language in profoundly deaf infants, Japan Journal of Logopedics and Phoniatrics 27 (1986) 235–243.

[26] M.A. Rodríguez, Lenguaje de signos, PhD. Dissertation, Confederación Nacional de Sordos Españoles (CNSE) and Fundación ONCE, Madrid. Spain, 1991.

[27] O. Juncos, A. Caamaño, MJ. Justo, E. López, RM. Rivas, MT. López, F. Sola, Primeras palabras en la lengua de signos española (LSE). Estructura formal, semántica y contextual, Dpto. Psicología Evolutiva, Facultad de Psicología, Universidad de Santiago de Compostela, Federación de Asociaciones de Sordos del País Gallego, 1996.

[28] IBM, web: 〈http://www.ibm.com/〉.

[29] Outsource-sl, web: 〈http://www.outsource-sl.com/fabricantes/IBM/ViaVoiceStd.htm〉.

[30] W. Ward, Extracting information from spontaneous speech, International Conference on Spoken Language Processing, September, 1994.

[31] W. Ward, B. Pellom, The CU Communicator System, in: Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Keystone, Colorado, 1999.

[32] W. Ward, B. Pellom, 2002. The Phoenix Parser User Manual, downloadable from 〈http://cslr.colorado.edu/beginweb/cumove/cucommunicator.html〉.

[33] Phoenix Parser Software, 〈http://cslr.colorado.edu/beginweb/cumove/cucommunicator.html〉.