Distributed Spanish Sign Language Synthesizer Architecture

Fernando López-Colino, Javier Tejedor, Javier Garrido and José Colás

Human Computer Technology Laboratory, Escuela Politécnica Superior,
Universidad Autónoma de Madrid, Spain
[fj.lopez, javier.tejedor, javier.garrido, jose.colas]@uam.es

Abstract. This work presents the design of a distributed Sign Language synthesis architecture. The main objective of this design is to adapt the synthesis process to the diversity of the user devices. The synthesis process has been divided into several independent modules that can be executed either in a synthesis server or in the client device. Depending on the modules assigned to the server or the client, four different scenarios have been defined. These scenarios may vary from a heavy client design which executes the whole synthesis, to a light client design similar to a video player. These four scenarios will provide the maximum signed message quality independently of the device hardware resources.

Key words: Spanish Sign Language, Ubiquitous Computing, Automatic Synthesis, Deaf People

1 Introduction

Literature provides several examples of Sign Language (SL) synthesizers [1, 2, 3]. The most extended technique is based on the animation of a virtual avatar based on abstract representations of the sign. These synthesizers require complex calculations, specific libraries and 3D capable devices, which reduces the suitable devices to PCs and laptops. Unfortunately, there are many real situations where only a mobile device is available.

Mobile device hardware resources do not fulfill current SL synthesizers' requirements. For this reason, instead of proposing a low quality SL synthesis adapted to these resources, we propose a distributed architecture. This architecture is divided into several modules that can be run either in a synthesis server or in the user's device, so the resulting message quality is always the same.

2 Distributed Architecture

The SL synthesizer has been designed to deal with the great diversity of final user devices. In order to cover most hardware and software platforms, a distributed architecture has been established, dividing the whole process into five different steps (Fig. 1): HLSML Parser, Avatar and Sign Description Retrieval, Gesture Synthesis, Rendering and Visualization. This synthesizer has been adapted to Spanish Sign Language (SpSL) contents.

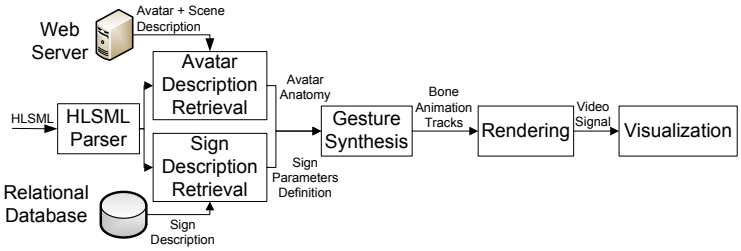

Fig. 1. This image shows the different elements of the SpSL Synthesizer.

HLSML Parser The sign message is described using a XML-based format. We have defined HLSML (High Level Signing Markup Language) to describe all the elements required for SpSL automatic synthesis. This module parses the HLSML message and obtains the sequence of SpSL signs.

Avatar and Sign Description Retrieval We have chosen the JSR-184 standard [4] for 3D scene description. This standard defines the m3g file format to describe all the elements in the scene such as geometry, lights, cameras, animation tracks, materials and hierarchical bone structures. The avatar, defined as a skeleton mesh with a multiple material assigned, the default lights and the four cameras are defined by means of this m3g file format. This file is downloaded from the Web Server using the standard HTTP protocol. The sign definitions have been recorded in a relational database by means of their basic parameters [5]. The database contains a standard sign representation and several variations for some basic sign parameters such as hand shapes and movements. The sign representation is different for every SL. In our case, this database contains the SpSL sign definitions.

Gesture Synthesis This is the main module of the whole synthesis process. Its role is to create the animation tracks for every skeleton bone that must be animated in the final sequence. Each animation track consists of a sequence of bone orientations with a timestamp that indicates the instant when the bone must reach that orientation. These orientations can be retrieved from the Relational Database, such as the orientation for the hand bones, or calculated using Inverse Kinematics, e.g. upper-arm and forearm orientations.

Rendering The rendering process consists of the generation of a 2D image from a 3D scene definition. The 3D animated scene definition is created by merging the scene definition retrieved from the Web Server and the bone animation tracks created in the Gesture Synthesis module. Each 2D image, i.e. each video frame, is generated at regular time intervals. These time intervals depend on the device hardware resources. The maximal time interval for a fluid animation is 0.06 s.

Visualization The visualization of the resulting video sequence is the last stage of the process, which defines two different alternatives. The first one involves performing the live visualization directly from the rendered output. The second alternative is reserved for devices with low 3D capabilities or without the required rendering API. In this case, the Visualization process would consist of playing the video sequence that can be downloaded after a synthesis server has finished rendering the whole message or while the message is being rendered by means of streaming technology.

3 Final User Device Adaptation

The most hardware dependent modules are the Gesture Synthesis, the Rendering and the Visualization. The first two modules can be assigned either to the final client device or to a synthesis server, while the Visualization must be run on the client side using one of the visualization alternatives described before. The distribution of these modules between server and client sides and the visualization alternative defines four different scenarios (Fig. 2):

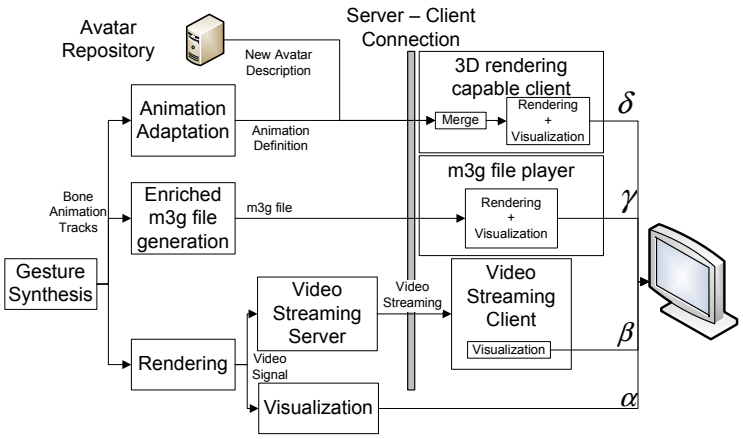

Fig. 2. This image shows the four scenarios designed for the final user device adaptation

- The first scenario (scenario α) is designed for PC devices. These devices have enough resources to run the whole synthesis process. For this reason all the modules are assigned to the client device. This solution does not involve a high network load because of the small size of the m3g scene description file and the reduced amount of data required in the sign definitions.
- The second scenario (scenario β) is suitable when the client device does not have 3D rendering resources or the required API is not available. This light client strategy requires only a video program on the client device. Obviously,

the main synthesis stages (Gesture Synthesis and Rendering) must be done on the server side. The resulting sequence is transferred to the client device by means of a standard file transfer protocol or by video streaming.
- The third is a low network load scenario that assigns the Gesture Synthesis module to the server side (scenario γ). Only a complete m3g file including an avatar description and bone animation tracks is transmitted to the client device. This requires only 3D graphic resources, so this third scenario is ideal for mobile devices with Java 3D mobile API. The downside of this scenario is that the server must generate the extended m3g file containing the scene description and the animation information.
- The last scenario (scenario δ) is similar to the third one as it is a low network load approach and requires 3D management capabilities in the client device. The bone animation tracks obtained in the Gesture Synthesis module are adapted and transmitted to the final user device. The client program merges these bone animation tracks with the avatar description in order to obtain the desired SpSL message. This scenario may be used as an interface to be used by other rendering technologies such as VRML, XNA, etc.

4 Current Development and Future Work

At the current state of the project, scenarios α and β are completely functional and they are under final user tests. The γ approach depends on the version 2.0 of the m3g library release which will implement the export feature for the γ scenario. Finally, the δ scenario is at the first stage of its development. We have chosen the VRML standard as the rendering technology to adapt our SpSL Synthesizer.

5 Acknowledgements

Authors would like to acknowledge the FPU-UAM program for its support.

References

[1] Zwiterslood, I., Verlinden, M., Ros, J., Schoot, S.: Synthetic signing for the deaf: esign. In: Proc. of the Conference and Workshop on Assistive Technologies for Vision and Hearing Impairment, Granada, Spain, CVHI (June 2004)
[2] Bangham, A., Cox, S., Elliot, R., Glauert, J., Marshall, I.: Virtual signing: Capture, animation, storage and transmission - an overview of the visicast project. In: IEE Sem. on Speech and Language Processing for Disabled and Elderly People. (2000)
[3] Kennaway, R., Glauert, J.R.W., Zwitserlood, I.: Providing signed content on the internet by synthesized animation. ACM Transactions on Computer-Human Interaction **14**(15) (2007) 1–29
[4] Java Community Process: Jsr-184. mobile 3d graphics api for j2me. http://www.jcp.org/en/jsr/detail?id=184 (2005)
[5] López, F., Tejedor, J., Garrido, J., Colás, J.: Use of a hierarchical skeleton for spanish sign language 3d representation over mobile devices. In: Proc. of INTERACCION, AIPO (November 2006) 565–568